TUGAS AKHIR – KS141501

PEMBUATAN TOOLS PEMANTAUAN KINERJA BASIS DATA SQL SERVER BERBASIS WEB (STUDI KASUS: PANTAU.ITS.AC.ID)

DEVELOPMENT OF WEB-BASE SQL SERVER DATABASE PERFORMANCE MONITORING TOOLS (CASE STUDY: PANTAU.ITS.AC.ID)

ERWIN WILBERT MAPALIEY
NRP 5212 100 094

Dosen Pembimbing
Radityo Prasetianto Wibowo, S.Kom.,M.Kom.

JURUSAN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KS141501

PEMBUATAN TOOLS PEMANTAUAN KINERJA BASIS DATA SQL SERVER BERBASIS WEB (STUDI KASUS: PANTAU.ITS.AC.ID)

ERWIN WILBERT MAPALIEY
NRP 5212 100 094

Dosen Pembimbing
Radityo Prasetianto Wibowo, S.Kom.,M.Kom.

JURUSAN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

FINAL PROJECT – KS141501

DEVELOPMENT OF WEB-BASE SQL SERVER DATABASE PERFORMANCE MONITORING TOOLS (CASE STUDY: PANTAU.ITS.AC.ID)

ERWIN WILBERT MAPALIEY
NRP 5212 100 094

Supervisor
Radityo Prasetianto Wibowo, S.Kom.,M.Kom.

INFORMATION SYSTEMS DEPARTMENT
Faculty of Information Technology
Tenth Nopember Institute of Technology
Surabaya 2017

LEMBAR PENGESAHAN

PEMBUATAN TOOLS PEMANTAUAN KINERJA BASIS DATA SQL SERVER BERBASIS WEB (STUDI KASUS: PANTAU.ITS.AC.ID)

TUGAS AKHIR
Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:

ERWIN WILBERT MAPALIEY
5212 100 094

Surabaya, Juli 2017

KEPALA
DEPARTEMEN SISTEM INFORMASI

Dr. Ir. Aris Tjahyanto, M.Kom.
NIP. 19650310 199102 1 001

LEMBAR PERSETUJUAN

PEMBUATAN TOOLS PEMANTAUAN KINERJA BASIS DATA SQL SERVER BERBASIS WEB (STUDI KASUS: PANTAU.ITS.AC.ID)

TUGAS AKHIR
Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:

ERWIN WILBERT MAPALIEY
5212 100 094

Disetujui Tim Penguji : Tanggal Ujian : 7 Juli 2017
Periode Wisuda : September 2017

Radityo Prasetianto W, S.Kom., M.Kom. **(Pembimbing I)**

Faizal Johan Atletiko, S.Kom., M.T. **(Penguji I)**

Renny Pradina Kusumawardani, S.T., M.T. **(Penguji II)**

PEMBUATAN TOOLS PEMANTAUAN KINERJA BASIS DATA SQL SERVER BERBASIS WEB (STUDI KASUS: PANTAU.ITS.AC.ID)

Nama Mahasiswa : Erwin Wilbert Mapaliey
NRP : 5212 100 094
Jurusan : Sistem Informasi
Dosen Pembimbing : Radityo Prasetianto W, S.Kom., M.Kom.

ABSTRAK

Basis data merupakan teknologi yang sudah menjadi bagian penting bagi suatu organisasi maupun pelaku bisnis. Basis data merupakan kumpulan data yang disimpan di dalam komputer secara sistematis sehingga mudah diakses, dikelola, dan diperbarui. Secara umum, terdapat tiga aktor utama dalam penggunaan basis data, yaitu administrator basis data (DBA), satu atau lebih pengembang aplikasi, dan pengguna. Seorang administrator basis data bertugas untuk mengatur seluruh aktivitas yang berkaitan dengan optimalisasi kinerja basis data.

Salah satu cara untuk menentukan apakah basis data sudah bekerja secara optimal adalah dengan melakukan pemantauan kinerja basis data. Beberapa vendor DBMS sudah menyediakan tools ini yang sudah termasuk di dalam produk yang mereka sediakan. Selain itu, ada pihak luar yang menyediakan tools pemantauan kinerja basis data ini secara terpisah dan mendukung multi-platform.

Insitut Teknologi Sepuluh Nopember (ITS) memiliki sebuah sistem berbasis web yang berfungsi untuk memantau seluruh aktivitas aplikasi yang digunakan di lingkungan ITS termasuk basis data, yang memiliki alamat pantau.its.ac.id. Akan tetapi belum terdapat fungsi pemantauan yang memberikan informasi terkait kinerja basis data. Penelitian ini bertujuan untuk

membuat tools pemantauan kinerja basis data berbasis web untuk diimplementasikan pada sistem pantau.its.ac.id sehingga memudahkan DPTSI sebagai pihak yang mengelola sistem basis data untuk mendapatkan informasi terkait kinerja basis data dan menentukan langkah yang harus dilakukan agar mendapatkan kinerja basis data yang efisien dan optimal.

Melalui proses pembuatan hingga penggujian didapatkan kesimpulan bahwa metrik-metrik yang digunakan dapat diimplementasikan pada sistem berbasis web, dan sistem yang dibuat dapat menyesuaikan dengan versi basis data yang berbeda.

Kata kunci: Basis Data, Pemantauan Kinerja, SQL Server, Performance Dashboard

DEVELOPMENT OF WEB-BASED SQL SERVER DATABASE PERFORMANCE MONITORING TOOLS (CASE STUDY: PANTAU.ITS.AC.ID)

Student Name : Erwin Wilbert Mapaliey
NRP : 5212 100 094
Department : Information Systems
Supervisoe : Radityo Prasetianto W, S.Kom., M.Kom.

ABSTRACT

Database is a technology that has become an important part for an organization or business. The database is a collection of data stored in the computer systematically so easily accessible, managed, and updated. In general, there are three main actors in database usage, the database administrators (DBA), one or more application developers, and users. A database administrator is tasked with managing all activities related to database performance optimization.

One way to determine whether the database is working optimally is to perform database performance monitoring. Some DBMS vendors already provide these tools that are included in their products. In addition, there are another company provide these database performance monitoring tools separately and support multi-platform.

Tenth Nopember Institute of Technology (ITS) has a web-based system that can monitor all application activities that used in ITS environment including the database, which has address pantau.its.ac.id. However, there is no monitoring function that provides information related to database performance. This final project aims to create a web-based database performance monitoring tool that can be implemented on pantau.its.ac.id so as to facilitate DPTSI as department that manages the database system to get information related to database performance and

determine the steps to be done in order to get the performance with efficient and optimal data usage.

Through the process of making up to the testers it can be concluded that the metrics used can be implemented on a web-based system, and the system created can adapt to different database versions

Keywords : Database, Performance Monitoring, SQL Server, Performance Dashboard

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas berkat rahmat dan penyertaan-Nya serta telah memberikan kesehatan, kekuatan dan kelancaran bagi penulis dalam menyelesaikan Tugas Akhir ini sebagai salah satu syarat kelulusan pada Jurusan Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Dengan judul penelitian tugas akhir yaitu:

"PEMBUATAN TOOLS PEMANTAUAN KINERJA BASIS DATA SQL SERVER BERBASIS WEB (STUDI KASUS: PANTAU.ITS.AC.ID)"

Dalam penyusunan, pengerjaan, dan penulisan Tugas Akhir ini tidak terlepas dari bantuan, bimbingan, serta dukungan dari berbagai pihak. Oleh karena itu, penulis dengan senang hati ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

- Keluarga penulis yang selalu memberikan doa, dukungan dan semangat selama saya menjalani kuliah di Sistem Informasi ITS.
- Bapak Dr. Ir. Aris Tjahyanto, M.Kom. sebagai Ketua Jurusan Sistem Informasi ITS dan Bapak Nisfu Asrul Sani, S.Kom, M.Sc. sebagai Kaprodi S1 Sistem Informasi ITS
- Bapak Dr. Eng. Febriliyan Samopa, S.Kom, M.Kom. sebagai dosen wali penulis selama menjalani perkuliahan di Jurusan Sistem Informasi ITS. Terima kasih untuk bimbingan dan saran yang membangun selama penulis menjalani perkuliahan
- Bapak dan ibu dosen laboratorium bidang minat Akuisisi Data dan Diseminasi Informasi.
- Bapak Radityo Prasetianto Wibowo, S.Kom., M.Kom. sebagai dosen pembimbing yang selalu sabar membimbing penulis selama pengerjaan dan penulisan Tugas Akhir ini. Terima kasih untuk waktu

yang telah diberikan untuk membimbing penulis, kritik dan saran yang membangun sehingga Tugas Akhir ini dapat diselesaikan dengan baik.

- Mas Ricky Asrul Sani sebagai admin dan laboran Lab. Akuisisi Data dan Diseminasi Informasi yang telah berbagi pengalaman kepada penulis selama menjadi Asisten Laboratorium dan mengatur jadwal Tugas Akhir dengan baik.
- Mas Rizky dari pihak DPTSI yang telah memberikan waktu untuk berdiskusi dan membantu penulis sehingga Tugas Akhir ini dapat diimplementasikan.
- Teman-teman seperjuangan bimbingan Pak Radit dan teman-teman ADDI AMAN yang telah berjuang bersama untuk menyelesaikan tugas akhir ini.
- SOLA12IS angkatan 2012 Jurusan Sistem Informasi yang selalu memberikan dukungan dan semangat.
- Semua pihak yang tidak dapat disebutkan satu-persatu yang juga telah banyak membantu dan memberikan semangat dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa penulisan laporan Tugas Akhir ini masih jauh dari sempurna. Oleh sebab itu, penulis memohon maaf atas segala kekurangan dan kesalahan yang ada di dalam Tugas Akhir ini. Penulis dengan senang hati menerima saran dan kritik untuk memperbaiki kekurangan dan kesalahan yang ada pada tugas akhir ini. Semoga tugas akhir ini dapat bermanfaat bagi pembaca.

Surabaya, Juli 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Pada bagian ini akan diuraikan gambaran umum penelitian yang meliputi latar belakang masalah, perumusan masalah, batasan masalah penelitian, dan relevansi terhadap bidang minat penelitian. Setelah menjelaskan tujuan dan manfaat penelitian diharapkan dapat memberikan gambaran awal mengenai penelitian yang akan dilakukan.

1.1. Latar Belakang Masalah

Penggunaan teknologi basis data saat ini sudah semakin banyak dan berkembang dari tahun ke tahun. Basis data adalah kumpulan data yang disimpan di dalam komputer secara sistematis sehingga kontennya dapat dengan mudah diakses, dikelola, dan diperbarui [1]. Tipe basis data yang paling populer saat ini adalah basis data relasional. Sebuah basis data relasional biasanya terdiri dari satu atau beberapa tabel atau relasi dari baris dan kolom dengan *unique key* yang mengidentifikasi tiap baris [2]. Terdapat sekurang-kurangnya tiga aktor utama dalam penggunaan basis data, yaitu administrator basis data (DBA), satu atau lebih pengembang aplikasi, dan pengguna [3]. Pada umumnya, seorang administrator basis data bertugas untuk mengatur seluruh aktivitas yang berkaitan dengan optimalisasi kinerja basis data.

Salah satu cara untuk menentukan apakah basis data yang digunakan sudah bekerja secara optimal adalah dengan melakukan pemantauan kinerja basis data [4]. Pemantauan kinerja basis data adalah tindakan untuk mengukur kinerja basis data secara *real-time* untuk menemukan masalah yang membuat kinerja basis data tidak optimal dan faktor-faktor lain yang dapat menyebabkan masalah di kemudian hari [6]. Dengan memantau kinerja basis data, seorang administrator basis data juga dapat menentukan area mana dari basis data yang dapat

ditingkatkan atau dioptimalkan untuk meningkatkan efisiensi dan kinerja.

Beberapa vendor DBMS memiliki tools untuk melakukan pemantauan kinerja dan biasanya sudah tersedia bersama dengan produk basis datanya, seperti Enterprise Management milik Oracle [7] dan SQL Server Performance Dashboard milik Microsoft [8]. Selain itu, ada pihak luar yang hanya menyediakan tools untuk memantau kinerja basis data dan mendukung basis data *multi-platform*, serta tersedia baik berbasis desktop maupun berbasis web seperti Database Health [9] dan ApexSQL [10]. Semua aplikasi tersebut memiliki fitur-fitur yang sangat baik namun untuk pemakaiannya tergantung kebutuhan dari setiap pengguna.

Institut Teknologi Sepuluh Nopember (ITS) memiliki sebuah sistem berbasis web yang berfungsi untuk memantau seluruh aktivitas aplikasi yang digunakan di lingkungan ITS termasuk basis data, yang memiliki alamat *pantau.its.ac.id* [11]. Sistem ini dibuat agar memudahkan DPTSI selaku pihak yang bertugas mengatur jalannya seluruh sistem teknologi informasi yang ada di ITS dalam melakukan pemantauan dan mengontrol apakah seluruh sistem berjalan dengan baik atau sudah tidak berfungsi, akan tetapi aplikasi ini masih perlu dikembangkan agar memiliki fungsi yang lebih lengkap salah satunya yaitu fungsi untuk melakukan pemantauan kinerja basis data yang terdapat di lingkungan ITS, dengan dilengkapi fitur *dashboard* yang dapat menampilkan informasi kinerja basis data secara visual.

Penelitian ini bertujuan untuk membuat tools pemantauan kinerja basis data berbasis web yang diharapkan dapat digunakan dan diimplementasikan pada situs *pantau.its.ac.id* berdasarkan metrik-metrik yang terdapat dalam SQL Server Performance Dashboard, dan mempelajari bagaimana informasi terkait kondisi server basis data diproses oleh SQL Server. Dengan adanya tools ini sistem *pantau.its.ac.id* yang telah ada diharapkan dapat berfungsi lebih baik dan memudahkan pihak

DPTSI dalam mengelola sistem teknologi informasi di lingkungan ITS, terutama dalam pengelolaan kinerja basis data agar kinerja basis data menjadi lebih efisien dan optimal.

1.2. Perumusan Masalah

Berdasarkan uraian latar belakang, maka rumusan permasalahan yang menjadi fokus dalam penelitian berikut antara lain:

1. Bagaimana membuat tools untuk melakukan pemantauan kinerja basis data berbasis web untuk sistem *pantau.its.ac.id* berdasarkan metrik-metrik pada SQL Server Performance Dashboard?

2. Bagaimana cara metrik-metrik pada SQL Server Performance Dashboard bekerja untuk memperoleh informasi terkait kondisi server basis data terkini?

3. Bagaimana cara menggunakan metrik-metrik pada SQL Server Performance Dashboard untuk diimplementasikan dalam bentuk web?

4. Apakah tools pemantauan kinerja basis data berbasis web yang dibuat berpengaruh terhadap versi SQL Server yang berbeda?

1.3. Batasan Masalah

Berdasarkan rumusan masalah yang telah disebutkan diatas, agar penelitian ini memperoleh hasil yang optimal, maka batasan masalah dalam penelitian ini antara lain:

1. Pembuatan tools pemantauan kinerja basis data berbasis web ini mengacu pada metrik-metrik pengukuran kinerja basis data di dalam SQL Server

Performance Dashboard diantaranya *Blocking, Missing Index*, dan *CPU Utilization*.

2. Tools *reporting* yang digunakan adalah SQL Server Performance Dashboard 2012.

3. DBMS yang akan dilakukan pemantauan kinerja adalah Microsoft SQL Server.

4. Hasil akhir dari penelitian ini adalah tools pemantauan kinerja basis data berbasis web yang siap diimplementasikan pada aplikasi sistem monitoring web ITS (*pantau.its.ac.id*).

1.4. Tujuan Penelitian

Berdasarkan perumusan masalah dan batasan masalah yang telah disebutkan diatas, maka tujuan akhir dari penelitian ini adalah membuat tools untuk melakukan pemantauan kinerja basis data berbasis web dengan mengacu pada metrik-metrik pengukuran kinerja basis data di dalam SQL Server Performance Dashboard yang kemudian dapat diimplementasikan pada sistem *pantau.its.ac.id*. Di dalam tools ini juga terdapat fitur *dashboard* yang dapat menampilkan informasi kinerja basis data secara visual berdasarkan metrik-metrik pengukuran yang digunakan.

1.5. Manfaat Penelitian

Manfaat yang diharapkan melalui penelitian ini adalah memudahkan pihak DPTSI selaku badan yang bertugas mengelola seluruh sistem aplikasi dan basis data yang terdapat di lingkungan ITS untuk memantau kinerja basis data yang dimiliki dan memberikan informasi-informasi secara visual yang akurat untuk menentukan langkah yang harus dilakukan agar basis data bekerja secara efisien dan optimal, serta

pemahaman lebih dalam tentang pemantauan kinerja basis data pada SQL Server.

1.6. Relevansi

Penelitian ini berkaitan dengan bidang penelitian pada Laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI) dan mata kuliah di Jurusan Sistem Informasi diantaranya Manajemen Admistrasi Basis Data, Desain Basis Data, Pemrograman Berbasis Web, Konstruksi dan Pengujian Perangkat Lunak, Interaksi Manusia dan Komputer, dan Sistem Pendukung Keputusan.

Halaman ini sengaja dikosongkan

BAB II
TINJAUAN PUSTAKA

Pada bagian ini akan dijelaskan mengenai beberapa penelitian sebelumnya dan beberapa dasar teori yang menjadi acuan dan landasan dalam melakukan penelitian ini. Landasan teori akan memberikan gambaran tentang konsep dan teknologi apa saja yang digunakan dalam penelitian ini.

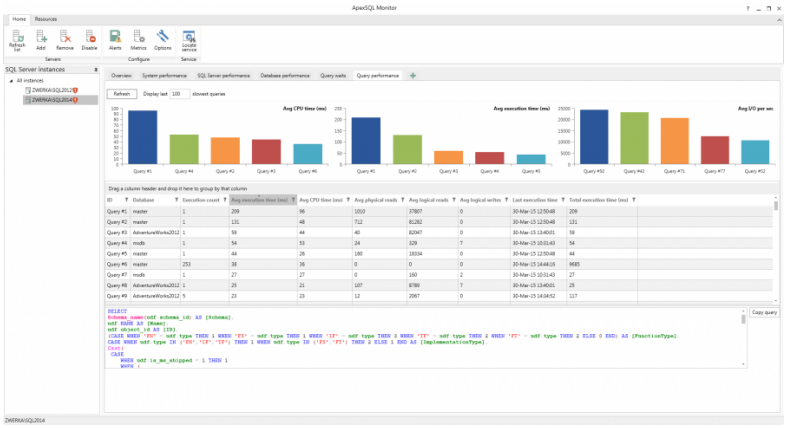
2.1. Penelitian Sebelumnya

Beberapa penelitian sebelumnya yang menjadi acuan dalam mengerjakan penelitian ini adalah sebagai berikut:

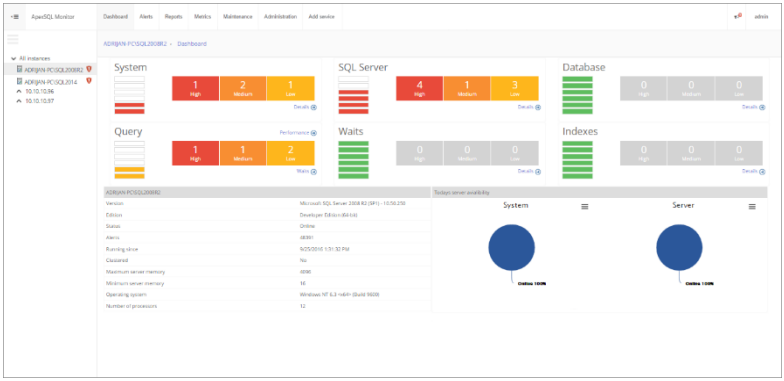
2.1.1. ApexSQL Monitor

ApexSQL Monitor merupakan salah satu aplikasi berbasis desktop buatan ApexSQL yang membantu seorang administrator basis data untuk melakukan pemantauan kinerja basis data SQL Server. Selain itu, ApexSQL Monitor juga dapat menampilkan metrik performa sistem basis data, mengidentifikasi masalah performa dan *deadlock*, dan menampilkan seluruh informasi performa basis data dalam bentuk *dashboard* sehingga administrator dapat melihat informasi metrik dan data performa basis data secara visual dan *real-time* [12].

Pada tahun 2017, ApexSQL Monitor mengeluarkan versi terbarunya yang berbasis web. Dalam versi terbaru ini juga menampilkan informasi yang berguna untuk melakukan pemantauan SQL Server Instance secara dinamis juga dengan tampilan *dashboard* yang sudah mengalami pembaruan desain yang memuat informasi yang penting seperti *alert* dan *availability* [13].



Gambar 2.1 Tampilan aplikasi ApexSQL Monitor berbasis desktop

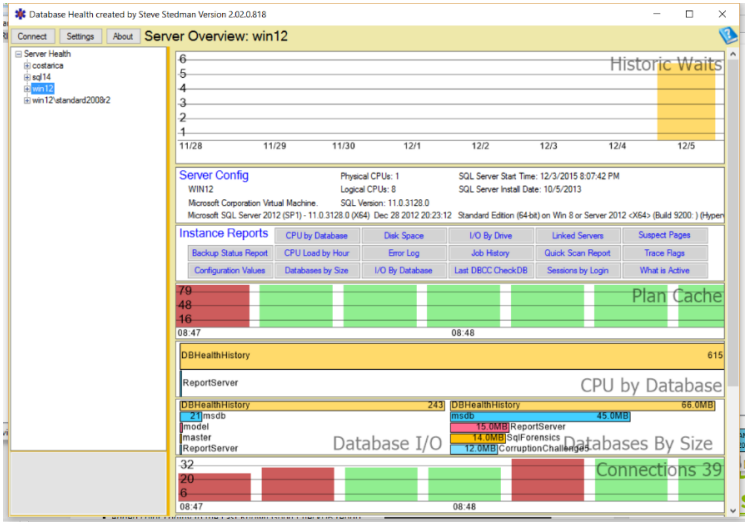


Gambar 2.2 Tampilan aplikasi ApexSQL Monitor 2017 berbasis web

2.1.2. Database Health Monitor

Database Health Monitor merupakan tools pemantauan kinerja basis data untuk SQL Server berbasis desktop. Tools ini dikembangkan oleh Steve Stedman pada tahun 2011 dengan mengacu pada SSRS Report sebagai dasar dalam pengembangannya. Tools ini berguna bagi administrator basis

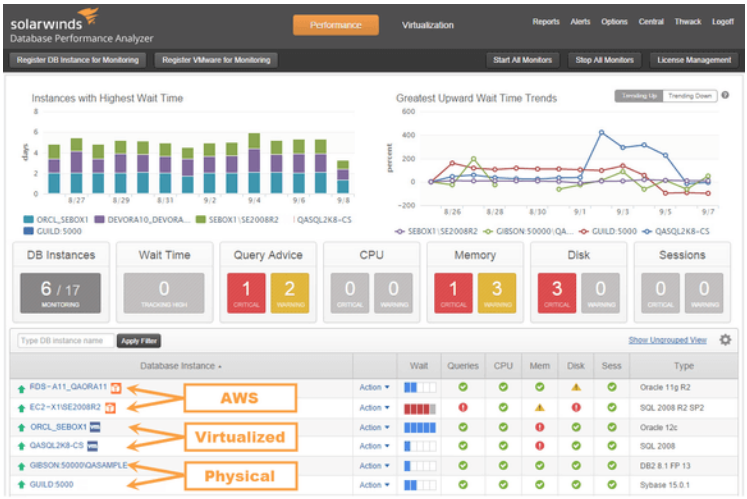
data yang membutuhkan tampilan visual untuk mengetahui bagaimana performa SQL Server Instance mereka [9].



Gambar 2.3 Tampilan aplikasi Database Health Monitor

2.1.3. Solarwinds Database Performance Analyzer

Solarwinds Database Performance Analyzer merupakan aplikasi monitoring basis data berbasis web buatan Solarwinds. Aplikasi ini mendukung pemantauan performa basis data *multi-platform*. Beberapa DBMS yang didukung diantaranya Microsoft SQL Server, Oracle, MySQL, IBM DB2, dan SAP ASE. Melalui aplikasi ini, administrator basis data dapat melihat performa basis data secara visual karena adanya fitur *dashboard*. Aplikasi ini juga memungkinkan seorang administrator basis data melakukan optimasi dimana saja dan kapan saja karena berbasis web dan mendukung *Cloud Monitoring* [14].



Gambar 2.4 Tampilan aplikasi Solarwinds Database Performance Analyzer

Dari ketiga aplikasi untuk melakukan pemantauan kinerja basis data tersebut yang dijadikan acuan untuk melakukan pembuatan tools pemantauan kinerja basis data, penulis melakukan perbandingan untuk melihat persamaan dan perbedaan antar aplikasi dengan tools pemantauan kinerja basis data yang akan dibuat dalam penelitian ini. Perbandingan antar aplikasi tersebut dapat dilihat pada tabel 2.1 berikut.

Tabel 2.1 Tabel perbandingan antar aplikasi

Aplikasi	Jenis Aplikasi	DBMS yang didukung	Fitur Dashboard
ApexSQL Monitor	Aplikasi Desktop, Aplikasi Web (2017)	Microsoft SQL Server	Ya
Database Health Monitor	Aplikasi Desktop	Microsoft SQL Server	Ya

Solarwinds Database Performance Analyzer	Aplikasi Web	Multi-platform	Ya
ITS Web-based Database Performance Monitoring	Aplikasi Web	Microsoft SQL Server	Ya

2.2. Dasar Teori

2.2.1. Database Performance Tuning & Monitoring

Database performance tuning & monitoring adalah serangkaian aktivitas yang dilakukan oleh administrator basis data untuk meningkatkan kinerja basis data agar lebih optimal. Sebelum melakukan penyetelan pada basis data, seorang administrator basis data harus memahami fungsi dari sebuah desain *physical database*, yaitu untuk menyimpan dan mengakses data dengan cara yang efisien. Terdapat beberapa faktor yang dapat digunakan untuk mengukur efisiensi kinerja basis data [15], yaitu:

- ***Transaction throughput***, merupakan rata-rata jumlah transaksi yang dapat diproses per menit oleh sistem basis data dan merupakan parameter kritis dari sistem transaksi (contoh : digunakan pada pemesanan tempat di pesawat, bank, dll). Hasil dari fase ini adalah penentual awal dari struktur penyimpanan dan jalur akses untuk file-file basis data.

- ***Response Time***, merupakan jumlah lama waktu yang dihitung dari akhir permintaan tersebut dilayani. Waktu yang telah berlalu dari suatu transaksi basis data yang diajukan untuk menjalankan suatu aksi. Dari sudut

pandang pengguna basis data, kita ingin memiminimalkan response time sedikit mungkin. Terdapat beberapa faktor yang mempengaruhi *response time* yang berada diluar jangkauan desainer basis data seperti *system loading* atau *communication times*.

- ***Disk storage***, merupakan jumlah ruang penyimpanan yang dibutuhkan untuk menyimpan suatu file basis data. Dalam hal ini seorang desainer basis data juga harus mengurangi jumlah penggunaan ruang penyimpanan.

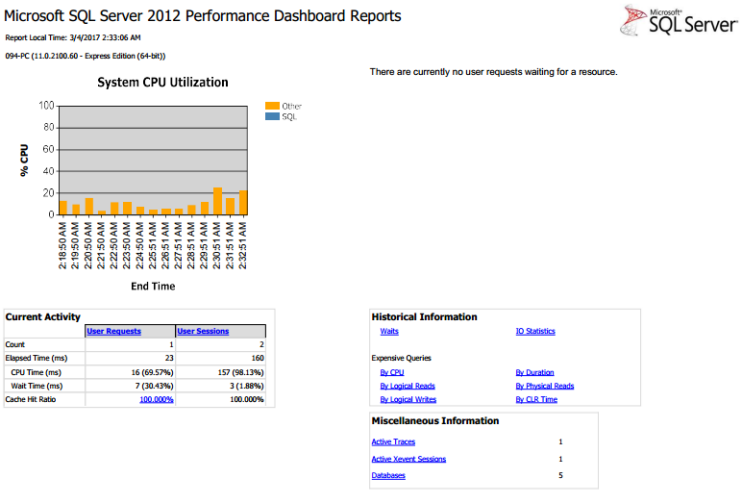
Namun, tidak ada satu faktor yang selalu benar. Seorang desainer basis data harus menimbang dan memperkirakan antara faktor satu dengan yang lainnya agar seimbang. Sebagai contoh, meningkatkan jumlah data yang disimpan akan mengurangi *response time* atau *transaction throughput*. Beberapa DBMS menyediakan tools bagi seorang administrator basis data untuk memantau dan melakukan penyetelan pada sistem.

Manfaat yang didapatkan dengan melakukan penyetelan basis data secara berkala adalah sebagai berikut:

- Mencegah pembelian hardware baru dengan tujuan meningkatkan kinerja sistem.
- Memungkinkan untuk menurunkan konfigurasi hardware.
- Dapat menghasilkan kinerja yang lebih cepat, yang menjadikan pengguna bekerja lebih produktif.
- *Response time* yang lebih singkat dapat meningkatkan moral pengguna.
- *Reponse time* yang lebih singkat dapat meningkatkan kepuasan pelanggan.

2.2.2. SQL Server Performance Dashboard

SQL Server Performance Dashboard merupakan proyek dashboard milik Microsoft yang berisikan laporan-laporan berdasarkan *Dynamic Management View* (DMV) yang digunakan untuk memberikan informasi secara visual terkait kinerja basis data SQL Server. SQL Server Performance Dashboard dapat menjadi tools yang berguna bagi seorang administrator basis data yang memerlukan pemantauan atau analisis lingkungan SQL Server [8].



Gambar 2.5 Tampilan SQL Server Performance Dashboard

Beberapa hal yang dapat dipantau dengan menggunakan tools SQL Server Performance Dashboard adalah sebagai berikut [8]:

- CPU
- SQL Server Memory Stats (PLE, Buffer cache ratio, Server memory, Buffer pool settings)
- Active user sessions
- Active Trace information

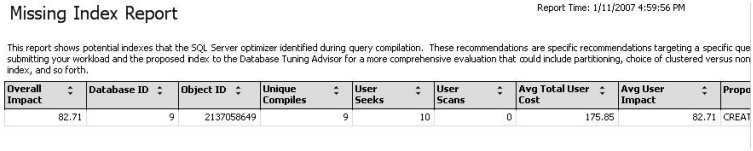
- General database information (Data File sizes, Log File sizes, Recovery Models etc)
- Historical waits stats
- Historical Data file IO
- Expensive queries (By CPU, By duration, By logical reads, By physical reads, By logical writes, CLR)
- Query plan stats (When Plan chaced, Showplan XML, Number of executions, Last execution times)
- Blocking information

Di dalam penelitian ini, penulis akan menggunakan metrik-metrik pengukuran kinerja basis data yang tersedia pada tools SQL Server Performance Dashboard diantaranya:

- ***Blocking***, merupakan suatu keadaan dimana terdapat satu *session* yang mengunci halaman sehingga tidak dapat diakses oleh *session* yang lain. Pada kasus tingkat tinggi, *Blocking* biasanya disebabkan oleh salah satu aplikasi atau *user* yang sedang menunggu SQL Server untuk menyelesaikan beberapa transaksi, atau SQL Server menunggu *client* untuk mengirimkan permintaan berikutnya, misal COMMIT atau permintaan ROLLBACK pada satu transaksi [16]. Dengan metrik ini, administrator dapat melihat *session* atau transaksi apa yang sedang berlangsung sehingga menyebabkan terjadinya *blocking* pada SQL Server.

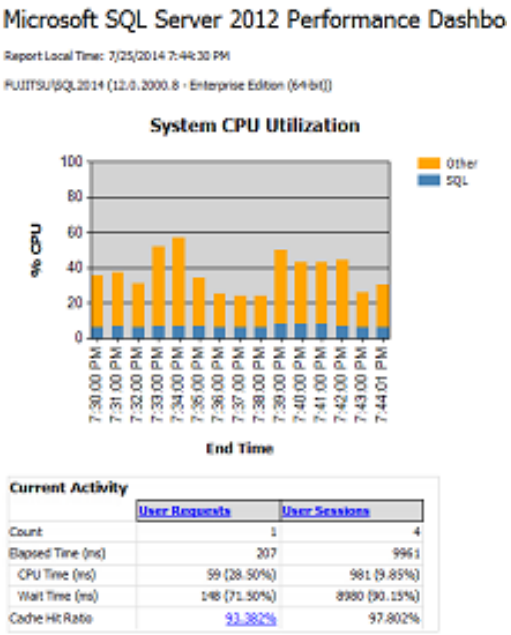
Gambar 2.6 Tampilan informasi *Blocking*

- ***Missing Index***, metrik ini berfungsi untuk menampilkan rekomendasi index pada sebuah tabel [16]. Index sangat mempengaruhi kinerja server basis data. Jika sebuah basis data dibuat tanpa menggunakan index, maka kinerja server dapat menurun drastis. Hal ini diakibatkan karena *resource* komputer banyak digunakan untuk pencarian data atau pengaksesan *query* SQL dengan metode *table-scan* sehingga membuat waktu eksekusi menjadi lebih lama.



Gambar 2.7 Tampilan informasi *Missing Index Report*

- ***CPU Utilization***, merupakan metrik yang menampilkan informasi tentang historis penggunaan CPU pada SQL Server dan juga dapat mendeteksi hal apa yang menyebabkan penggunaan CPU yang tinggi, misalnya *query* yang banyak pada satu transaksi. Konsumsi sumber daya CPU yang besar dapat menyebabkan kinerja server menjadi lambat dan tidak optimal.

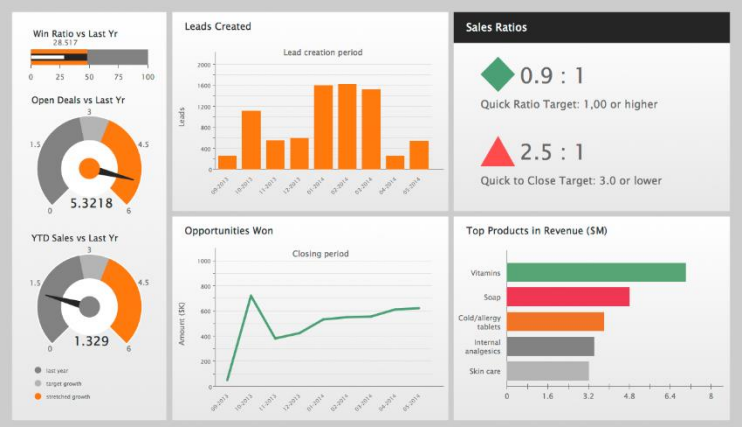


Gambar 2.8 Tampilan informasi penggunaan CPU pada SQL Server

Metrik-metrik ini tersimpan dalam format .rdl yang didapatkan ketika menginstal SQL Server Performance Dashboard dan untuk mengaksesnya yaitu dengan memanggil *stored procedure* yang terdapat di dalam file .rdl tersebut. *Query* dalam *stored procedure* tersebut yang akan digunakan pada pembuatan tools pemantauan kinerja basis data berbasis web.

2.2.3. Dashboard

Menurut Stephen Few [17], *Dashboard* merupakan tampilan visual dari informasi yang paling penting dan dibutuhkan untuk mencapai satu tujuan atau lebih, informasi visual diatur pada satu halaman sehingga dapat dipantau sekilas. Informasi pada sebuah *dashboard* disajikan secara visual, dengan menggunakan kombinasi teks dan grafis, tetapi dengan penekanan pada grafis. Sebuah *dashboard* dibuat secara visual agar dapat menyampaikan informasi yang lebih efisien dan memiliki makna yang sangat kaya daripada teks biasa.



Gambar 2.9 Contoh *dashboard* untuk menampilkan *Performance Indicator*

Pada umumnya, desain *dashboard* menggunakan tipe grafik. Data-data kuantitatif akan ditampilkan dalam bentuk grafik dua dimensi dengan sumbu X dan Y. Beberapa tipe grafik *dashboard* yang paling sering digunakan adalah sebagai berikut:

- *Bar graphs (horizontal and vertical)*
- *Stacked bar graphs (horizontal and vertical)*
- *Kombinasi bar dan line graphs*
- *Line graphs*
- *Sparklines*

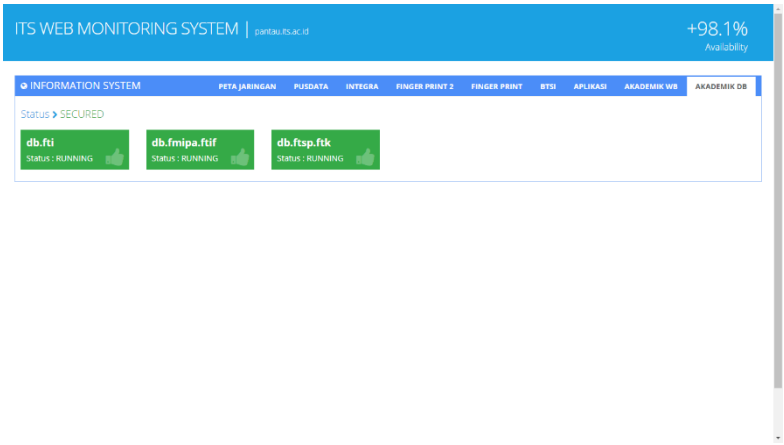
- *Box plots*
- *Scatter plots*
- *Treemaps*

Pada penelitian ini, *dashboard* akan digunakan sebagai media penyampaian informasi visual untuk melihat kinerja basis data. Dengan adanya fitur *dashboard* diharapkan dapat memudahkan administrator basis data untuk mendapatkan informasi secara cepat dan efisien sehingga dapat segera mengambil tindakan dengan cepat pula apabila terdapat masalah pada kinerja basis data yang sedang dipantau.

2.2.4. ITS Web Monitoring System

ITS Web Monitoring System merupakan sebuah aplikasi milik Institut Teknologi Sepuluh Nopember yang memiliki fungsi untuk melakukan pemantauan terhadap seluruh sistem aplikasi dan basis data yang terdapat di lingkungan ITS. ITS Web Monitoring System ini memiliki alamat *pantau.its.ac.id*. Saat ini, ITS Web Monitoring System hanya mampu memberikan informasi apakah suatu sistem sedang aktif atau tidak [11].

Gambar 2.10 Tampilan peta jaringan pada situs *pantau.its.ac.id*



Gambar 2.11 Halaman informasi basis data yang akan dilengkapi fitur *monitoring dashboard*

Melalui penelitian ini diharapkan dapat memperlengkapi fitur pemantauan kinerja basis data pada sistem ini dan membantu DPTSI selaku administrator jaringan di lingkungan ITS untuk mendapatkan informasi terkait kinerja basis data secara *real-time* dan dapat mengambil keputusan agar kinerja basis data tetap efisien dan optimal.

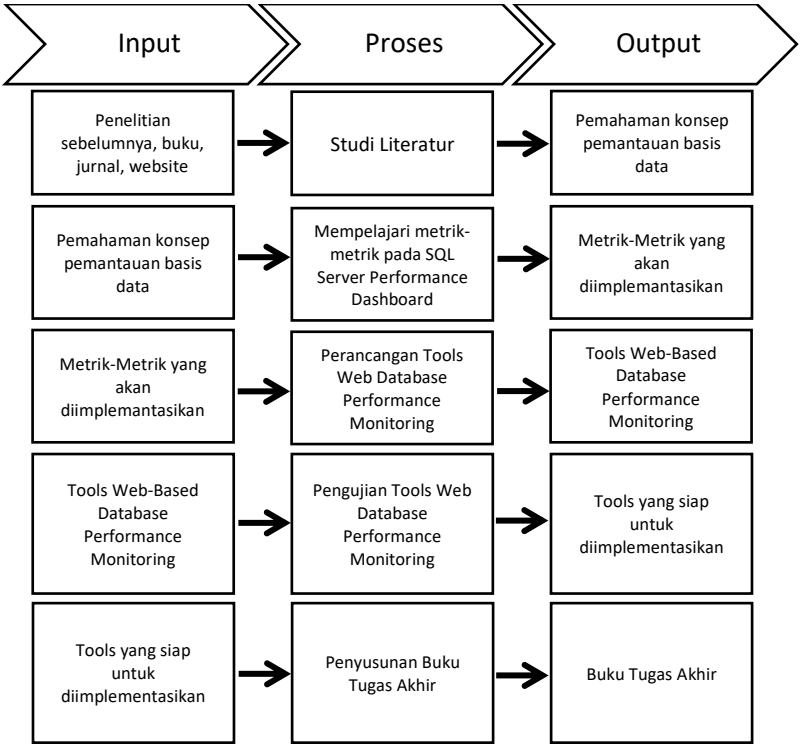
Halaman ini sengaja dikosongkan

BAB III
METODOLOGI PENELITIAN

Pada bagian metode penelitian akan dijelaskan mengenai tahapan-tahapan apa saja yang akan dilakukan dalam penelitian ini beserta deskripsi dan penjelasan dari tiap tahapan tersebut.

3.1. Tahapan Pelaksanaan Tugas Akhir

Bagian ini akan menjelaskan mengenai metodologi penelitian yang akan dilakukan. Tahapan-tahapan yang akan dilalui seperti pada Gambar 3.1



Gambar 3.1 Metodologi Penelitian

3.1.1. Studi Literatur

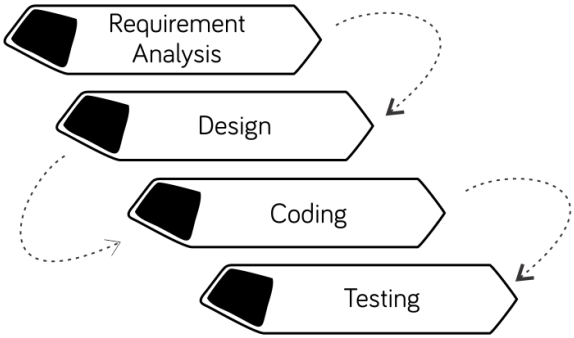
Pada tahap ini, penulis akan melakukan pengumpulan literatur yang berkaitan dengan penelitian ini. Literatur yang dikumpulkan merupakan hasil penelitian sebelumnya yang pernah dilakukan yang terkait pemantauan kinerja basis data dan beberapa situs web yang menyediakan informasi tentang topik pemantauan kinerja basis data. Hasil yang didapatkan pada tahap ini adalah pemahaman tentang konsep pemantauan kinerja basis data secara umum.

3.1.2. Mempelajari Metrik-Metrik SQL Server Performance Dashboard

Pada tahap ini, penulis akan fokus mempelajari tools SQL Server Performance Dashboard. Pada tools ini terdapat beberapa metrik pengukuran kinerja basis data yang digunakan. Penelitian ini berfokus pada beberapa metrik yang akan diimplementasikan pada sistem *pantau.its.ac.id* diantaranya *Blocking, Missing Index*, dan *CPU Utilization*. Hasil yang didapatkan pada tahap ini adalah pemahaman tentang metrik-metrik yang akan dipergunakan pada tools pemantauan kinerja basis data berbasis web yang akan dikerjakan dan menampilkan hasil informasi berdasarkan metrik tersebut secara visual.

3.1.3. Perancangan Tools Pemantauan Kinerja Basis Data Berbasis Web

Pada tahap ini, akan dilakukan perancangan dan pengembangan tools pemantauan kinerja basis data yang merupakan hasil dari analisa kebutuhan tools pemantauan kinerja basis data. Tahapan yang dilakukan untuk merancang dan mengembangkan tools pemantauan kinerja basis data ini menggunakan metode pengembangan perangkat lunak *Waterfall*.



Gambar 3.2 Alur pengembangan aplikasi dengan metode *Waterfall*

Adapun tahapan-tahapan dalam perancangan dan pengembangan perangkat lunak dengan metode *Waterfall* adalah sebagai berikut:

1. Perancangan
   Pada tahap ini, penulis akan membuat perancangan tentang gambaran tools pemantauan kinerja basis data yang akan dibuat. Di dalam tahap perancangan, pertama-tama penulis akan menganalisis kebutuhan fungsional sistem pemantauan kinerja basis data, siapa saja pengguna yang akan menggunakan tools ini, dan merancang alur kerja sistem yang kemudian akan digunakan sebagai acuan pada tahapan selanjutnya yaitu desain sistem.

2. Desain
   Setelah melakukan perancangan, pada tahap ini dilakukan beberapa proses desain antara lain:
   a. Desain Arsitektur Sistem
   b. Desain Basis Data

c. Desain Antarmuka Sistem
d. Desain Crawler

3. Pengkodean
   Tahapan ini adalah tahap implementasi desain yang telah dibuat pada tahapan sebelumya menjadi sebuah kode program untuk menghasilkan sebuah sistem. Sistem pemantauan kinerja basis data ini dikembangkan dengan bahasa pemrograman web PHP, MySQL sebagai sistem manajemen basis data, dan SQL Server untuk melakukan pemanggilan *stored procedure* pada SQL Server Performance Dashboard.

4. Pengujian
   Tahapan pengujian merupakan tahapan terakhir dalam rangkaian proses pembuatan tools pemantauan kinerja basis data ini. Pada tahap ini akan dilakukan pengujian terhadap sistem apakah sudah sesuai dengan kebutuhan yang telah dirancangkan sebelumnya. Jika sistem sudah memenuhi kebutuhan sistem yang telah dibuat sebelumnya, maka pada tahap selanjutnya akan dilakukan pengujian dengan versi basis data yang berbeda-beda.

3.1.4. Pengujian Tools Pemantauan Kinerja Basis Data

Setelah membuat tools pemantauan kinerja basis data berbasis web, pada tahap ini penulis akan melakukan pengujian terhadap tools yang telah dibuat. Pengujian dilakukan dengan mencoba melakukan pemantauan server basis data dengan versi SQL Server yang berbeda-beda. Hal ini dilakukan karena pada lingkungan server basis data ITS terdapat versi SQL Server yang beragam. Hasil yang didapatkan pada tahap ini yaitu Tools Pemantauan Kinerja Basis Data yang siap untuk diimplementasikan pada sistem *pantau.its.ac.id*.

3.1.5. Penyusunan Buku Tugas Akhir

Pada tahap ini, penulis akan melakukan dokumentasi keseluruhan proses penelitian yang dikemas dalam buku tugas akhir. Penulisan buku tugas akhir dilakukan dengan metode penulisan ilmiah.

Halaman ini sengaja dikosongkan

BAB IV
PERANCANGAN

Pada bab ini, akan dilakukan analisis dan perancangan terhadap sistem pemantauan kinerja basis data berbasis web. Dalam bab perancangan ini akan dijelaskan tentang proses analisis kebutuhan dan desain yang akan diimplementasikan pada sistem.

4.1. Analisis Kebutuhan

Analisis kebutuhan dilakukan berdasarkan fungsi utama perangkat lunak dengan pengguna sistem, kemudian dihasilkan daftar kebutuhan fungsional sistem yang akan dibuat selanjutnya.

4.1.1. Fungsi Utama Perangkat Lunak

Pembuatan tools pemantauan kinerja basis data SQL Server berbasis web ini bertujuan untuk membantu pengelola sistem teknologi informasi di ITS dalam memantau kondisi server basis data yang ada saat ini. Sistem pemantauan kinerja basis data ini dapat memberikan informasi secara visual mengenai beberapa kondisi server basis data, misalnya apabila terjadi *Blocking* yang membuat kinerja basis data menjadi lama, maka pihak pengelola dapat segera mengetahui penyebab terjadinya gangguan tersebut dan dapat segera mengambil tindakan yang tepat.

4.1.2. Pengguna Sistem

Berikut adalah pengguna sistem pemantauan kinerja basis data dan aktivitas yang dilakukan oleh pengguna sistem.

Tabel 4.1 Daftar pengguna sistem dan aktivitas yang dilakukan

Pengguna	Deskripsi	Aktivitas pada sistem
DPTSI	Merupakan pihak yang bertanggung jawab untuk mengelola jalannya sistem teknologi informasi di lingkungan ITS termasuk basis data	• Melihat kondisi terkini server basis data • Melihat informasi penggunaan CPU pada server basis data • Melihat informasi *Blocking* pada server basis data • Melihat informasi apabila terdapat *Missing index* pada salah satu basis data yang terdapat di server basis data • Melihat historis kondisi server basis data pada waktu tertentu • Menambah, mengurangi, memperbarui server basis data pada sistem

4.1.3. Kebutuhan Fungsional

Berdasarkan fungsi utama perangkat lunak dan pengguna sistem serta aktivitas yang dilakukan pada sistem, maka kebutuhan fungsional untuk tools pemantauan kinerja basis data ini antara lain:

- Sistem dapat menampilkan kondisi server basis data saat ini.
- Sistem dapat menampilkan informasi penggunaan CPU pada server basis data.
- Sistem dapat menampilkan informasi jika terjadi *Blocking* pada server basis data.
- Sistem dapat menampilkan informasi *Missing Index* yang terdeteksi pada server basis data.
- Sistem dapat menampilkan informasi historis keadaan CPU pada waktu tertentu.
- Sistem dapat menampilkan informasi historis jika terjadi *Blocking* pada waktu tertentu.
- Sistem dapat menampilkan informasi historis jika terdeteksi *Missing Index* pada waktu tertentu

Sedangkan kebutuhan non fungsional pada sistem yaitu sistem dapat memudahkan administrator basis data untuk menambah, mengurangi, dan memperbarui data server basis data.

4.2. Desain Sistem

Berikut adalah penjelasan tentang desain sistem yang akan digunakan untuk membuat tools pemantauan kinerja basis data berbasis web. Desain sistem yang akan dijelaskan meliputi metrik-metrik yang akan digunakan, desain arsitektur sistem, desain basis data, desain crawler, dan desain antarmuka sistem.

4.2.1. Metrik-Metrik SQL Server Performance Dashboard yang Digunakan

Berdasarkan penjelasan tentang metrik-metrik SQL Server Performance Dashboard yang akan digunakan pada bab sebelumnya, terdapat 3 metrik yang akan digunakan pada tools pemantauan kinerja basis data berbasis web ini. Metrik-metrik tersebut adalah *Blocking*, *Missing Index*, dan *CPU Utilization.*

Penjelasan masing-masing metrik beserta cara penggunaannya adalah sebagai berikut.

4.2.1.1. Blocking

Blocking merupakan suatu keadaan dimana terdapat satu *session* yang mengunci halaman sehingga tidak dapat diakses oleh *session* yang lain. Pada kasus tingkat tinggi, *Blocking* biasanya disebabkan oleh salah satu aplikasi atau user sedang menunggu SQL Server untuk menyelesaikan beberapa transaksi, misalnya UPDATE atau INSERT, atau SQL Server menunggu *client* untuk mengirimkan permintaan berikutnya, misal COMMIT atau permintaan ROLLBACK pada satu transaksi [16].

Stored procedure yang digunakan untuk mengambil data-data yang memuat informasi tentang adanya *blocking* pada server yaitu **MS_PerfDashboard.usp_Blocking**. Berdasarkan query yang digunakan, *stored procedure* ini mengambil data dari DMV (Dynamic Management Views) diantaranya **sys.dm_exec_requests**, **sys.dm_exec_sessions**, **sys.dm_tran _session_transactions**, dan **sys.dm_tran_active _transactions**.

- ***sys.dm_exec_requests*** merupakan DMV yang memuat informasi tentang setiap request yang sedang dijalankan dalam instance SQL Server. Jika terdapat request pada instance SQL Server, request tersebut langsung tercatat pada DMV ini. Untuk tujuan troubleshooting, DMV ini dapat digunakan untuk membantu mengidentifikasi kueri yang telah berjalan lama [19][21].

- ***sys.dm_exec_sessions*** merupakan DMV yang memuat informasi tentang setiap session atau pengguna yang telah terotentikasi oleh server. DMV akan mencatat status dan aktivitas session baru pada instance SQL Server [19][20].

- ***sys.dm_tran_active_transactions*** merupakan DMV yang memuat informasi dan mencatat transaksi yang sedang aktif saat ini pada instance SQL Server [22].

- ***sys.dm_tran_session_transaction*** merupakan DMV yang memuat informasi dan mencatat transaksi yang sedang dilakukan oleh session [23].

Query yang digunakan pada *stored procedure* **MS_PerfDashboard.usp_Blocking** adalah sebagai berikut:

```
1.  BEGIN
2.     WITH blocking_hierarchy (head_wait_resource, ses
    sion_id, blocking_session_id, tree_level, request_i
    d, transaction_id, status, sql_handle, plan_handle,
     statement_start_offset, statement_end_offset, wait
    _type, wait_time, wait_resource, program_name, seco
    nds_active_idle, open_transaction_count, transactio
    n_isolation_level)
3.     AS (
4.  SELECT
5.     (SELECT MIN(wait_resource)
6.     FROM sys.dm_exec_requests WHERE blocking_session
       _id = s.session_id) AS head_wait_resource,
7.     s.session_id,
8.     convert(smallint, NULL),
9.     convert(int, 0), r.request_id,
10.    coalesce(r.transaction_id, st.transaction_id),
11.    isnull(r.status, 'idle'), r.sql_handle, r.plan_h
       andle, r.statement_start_offset, r.statement_end
       _offset, r.wait_type, r.wait_time, r.wait_resour
       ce, s.program_name,
12.    case when r.request_id is null then datediff(ss,
     s.last_request_end_time, getdate()) else datediff(
    ss, r.start_time, getdate()) end,
13.    convert(int, p.open_tran),
14.    coalesce(r.transaction_isolation_level, s.transa
    ction_isolation_level)
15.    FROM sys.dm_exec_sessions s
16.    JOIN sys.sysprocesses p ON s.session_id = p.spid
```

```
17.     LEFT JOIN sys.dm_exec_requests r ON s.session_id
     = r.session_id
18.     LEFT JOIN sys.dm_tran_session_transactions st ON
     s.session_id = st.session_id
19.     WHERE s.session_id IN (select blocking_session_i
    d from sys.dm_exec_requests)
20.     AND isnull(r.blocking_session_id, 0) = 0
21.
22.     UNION ALL
23.
24.     SELECT b.head_wait_resource, r.session_id, r.blo
    cking_session_id, tree_level + 1, r.request_id, r.t
    ransaction_id, r.status, r.sql_handle, r.plan_handl
    e, r.statement_start_offset, r.statement_end_offset
    , r.wait_type, r.wait_time, r.wait_resource, NULL,N
    ULL, r.open_transaction_count, r.transaction_isolat
    ion_level
25.     FROM sys.dm_exec_requests r
26.     JOIN blocking_hierarchy b ON r.blocking_session_
    id = b.session_id
27.     )
28.     SELECT b.head_wait_resource, b.session_id, b.req
    uest_id, b.blocking_session_id, b.program_name, b.t
    ree_level,
29.     case when LEN(qt.query_text) < 2048 then qt.quer
    y_text else LEFT(qt.query_text, 2048) + N'...' end
    AS query_text,
30.   master.dbo.fn_varbintohexstr(b.sql_handle) AS sql
    _handle,
31.   master.dbo.fn_varbintohexstr(b.plan_handle) AS pl
    an_handle,
32.     b.statement_start_offset,
33.     b.statement_end_offset,
34.     b.status AS session_or_request_status,
35.     b.wait_type, b.wait_time, b.wait_resource, b.tra
        nsaction_id, b.transaction_isolation_level, b.op
        en_transaction_count, b.seconds_active_idle,
36.     t.name AS transaction_name, t.transaction_begin_
        time, t.transaction_type, t.transaction_state, t
        .dtc_state, t.dtc_isolation_level, st.enlist_cou
        nt, st.is_user_transaction, st.is_local, st.is_e
        nlisted, st.is_bound
37.     FROM blocking_hierarchy b
```

```
38.     LEFT JOIN sys.dm_tran_session_transactions st ON
     st.transaction_id = b.transaction_id AND st.sessio
    n_id = b.session_id
39.     LEFT JOIN sys.dm_tran_active_transactions t ON t
    .transaction_id = b.transaction_id
40.     OUTER apply msdb.MS_PerfDashboard.fn_QueryTextFr
    omHandle(b.sql_handle, b.statement_start_offset, b.
    statement_end_offset) AS qt
41. END
```

Gambar 4.1 Query pada *stored procedure* MS_PerfDashboard.usp_Blocking

Jika *query* tersebut dijalankan, maka akan menghasilkan tabel dengan kolom-kolom yang memuat informasi tentang blocking pada server basis data. Namun, tidak semua kolom akan digunakan pada sistem. Berikut kolom-kolom yang akan digunakan pada sistem yang dibuat seperti pada Tabel 4.2 [20][21][22][23] dengan penjelasan sebagai berikut:

Tabel 4.2 Daftar kolom pada *MS_PerfDashboard.usp_Blocking* yang digunakan

Nama Kolom	Deskripsi	Tipe Data	Sumber DMV
session_id	ID Session yang menyebabkan terjadi *blocking* pada server basis data.	smallint	sys.dm_exec_sessions, sys.dm_exec_requests
request_id	Mengidentifikasi ID request yang sedang berjalan.	int	sys.dm_exec_requests
blocking_session_id	ID Session yang menyebabkan *blocking* untuk sementara waktu.	smallint	sys.dm_exec_requests

session_or_request_status	Mengidentifikasi status pada permintaan session. Status tersebut dapat berupa: - *Background* - *Running* - *Runnable* - *Sleeping* - *Suspended*	nvarchar (30)	sys.dm_exec_sessions, sys.dm_exec_requests
transaction_name	Mengidentifikasi nama jenis transaksi oleh session.	nvarchar (32)	sys.dm_tran_active_transactions
transaction_isolation_level	Mengidentifikasi level keberhasilan isolasi transaksi oleh session. 0 = Unspecified 1 = Read Uncomitted 2 = Read Committed 3 = Repeatable 4 = Serializable 5 = Snapshot	smallint	sys.dm_exec_requests
wait_type	Mengidentifikasi tipe *locking* yang sedang dialami oleh session yang sedang menunggu transaksi.	nvarchar (60)	sys.dm_exec_requests

wait_time	Jumlah waktu tunggu session yang sedang mengalami *blocking*.	int	sys.dm_exec_requests
wait_ resource	Mengidentifikasi sumber yang permintaannya saat itu sedang menunggu.	nvarchar (256)	sys.dm_exec_requests
query_text	*Query* pengguna/session yang mengalami *blocking*.	varbinary (64)	sys.dm_exec_requests

4.2.1.2. Missing Index

Missing Index berfungsi untuk menampilkan rekomendasi index pada sebuah tabel [16]. Index sangat mempengaruhi kinerja server basis data. Jika sebuah basis data dibuat tanpa menggunakan index, maka kinerja server dapat menurun drastis. Hal ini diakibatkan karena resource komputer banyak digunakan untuk pencarian data atau pengaksesan query SQL dengan metode *table-scan* sehingga membuat waktu eksekusi menjadi lebih lama.

Stored procedure yang digunakan untuk mengambil data-data yang memuat informasi tentang missing index yang terdeteksi pada server yaitu MS_PerfDashboard.usp_MissingIndexs dan MS_PerfDashboard.usp_MissingIndexStats. Namun untuk pembuatan sistem pada penelitian ini, stored procedure yang digunakan yaitu **MS_PerfDashboard.usp_MissingIndexStats** karena memuat informasi yang lebih detail terkait missing index pada server. Berdasarkan query yang digunakan, stored procedure ini mengambil data dari DMV (Dynamic Management Views) diantaranya **sys.dm_db_missing_index_**

groups, **sys.dm_db_missing_index_group_stats**, dan **sys.dm_db_missing_index_details**

- ***sys.dm_db_missing_index_details*** merupakan DMV yang memuat dan mencatat informasi detail tentang adanya missing index pada instance SQL Server, namun tidak termasuk index spasial [25].

- ***sys.dm_db_missing_index_groups*** dan ***sys.dm_db_missing_index_group_stats*** merupakan DMV yang memuat dan mencatat informasi tentang grup missing index yang terdeteksi pada instance SQL Server. DMV ini juga mencatat rata-rata dampak dari index yang berpengaruh pada server [24].

Query yang digunakan pada *stored procedure* **MS_PerfDashboard.usp_MissingIndexStats** adalah sebagai berikut:

```
1.  BEGIN
2.  SELECT d.database_id, d.object_id, d.index_handle,
    d.equality_columns, d.inequality_columns, d.include
    d_columns, d.statement AS fully_qualified_object,
3.    gs.*
4.  FROM sys.dm_db_missing_index_groups g
5.  JOIN sys.dm_db_missing_index_group_stats gs ON gs.g
    roup_handle = g.index_group_handle
6.  JOIN sys.dm_db_missing_index_details d ON g.index_h
    andle = d.index_handle
7.  WHERE d.database_id = isnull(@DatabaseID , d.databa
    se_id) AND d.object_id = isnull(@ObjectID, d.object
    _id)
8.  END
```

Gambar 4.2 Query pada *stored procedure* MS_PerfDashboard.usp_MissingIndexStats

Jika dijalankan, *query* tersebut akan menghasilkan tabel dengan kolom-kolom yang memuat informasi tentang *Missing Index* yang terdeteksi pada server basis data. Namun tidak semua

kolom akan ditampilkan pada sistem yang akan dibuat. Tabel 4.3 [24][25] menunjukan kolom-kolom yang akan digunakan pada sistem yang akan dibuat dalam penelitian ini berikut penjelasannya:

Tabel 4.3 Daftar kolom pada *MS_PerfDashboard.usp_MissingIndexStats* yang digunakan

Nama Kolom	Deskripsi	Tipe Data	Sumber DMV
database_id	Mengindentifikasi basis data dimana terdapat tabel yang terdeteksi *missing index.*	smallint	sys.dm_db_missing_index_details
object_id	Mengidentifikasi tabel dimana terdapat index yang hilang	int	sys.dm_db_missing_index_details
unique_compiles	Jumlah kompilasi dan rekompilasi yang akan diuntungkan dari kelompok indeks yang hilang. Kompilasi dan rekompilasi dari banyak pertanyaan berbeda dapat berkontribusi pada nilai kolom.	bigint	sys.dm_db_missing_index_group_stats

user_seeks	Jumlah pencarian yang disebabkan oleh *query* pengguna, yang menyebabkan rekomendasi index harus digunakan	bigint	sys.dm_db_ missing_ index_group_ stats
user_scans	Jumlah *scan* yang disebabkan oleh *query* pengguna, yang menyebab-kan rekomendasi index harus digunakan	bigint	sys.dm_db_ missing_ index_group_ stats
avg_total_ user_cost	Rata-rata jumlah biaya yang akan dapat dikurangi dengan meng-gunakan grup index yang di-rekomendasikan	float	sys.dm_db_ missing_ index_group_ stats
avg_user_ impact	Rata-rata persen-tase keuntungan pada *user query* jika rekomendasi index di-implementasikan.	float	sys.dm_db_ missing_ index_group_ stats

avg_system_impact	Rata-rata persentase keuntungan pada *system query* jika rekomendasi index di-implementasikan.	float	sys.dm_db_missing_index_group_stats
index_handle	Mengidentifikasi indeks yang hilang milik kelompok yang ditentukan oleh index_group_handle.	int	sys.dm_db_missing_index_groups, sys.dm_db_missing_index_details
fully_qualified_object	Nama tabel yang kehilangan index	nvarchar (4000)	sys.dm_db_missing_index_details
equality_columns	Daftar kolom yang dipisahkan koma, yang berkontribusi terhadap predikat persamaan: *table.column = constant_value*	nvarchar (4000)	sys.dm_db_missing_index_details

inequality_ columns	Daftar kolom yang dipisahkan koma yang berkontribusi terhadap ketidaksetaraan predikat, misalnya, predikat bentuk: *table.column* > *constant_value*	nvarchar (4000)	sys.dm_db_ missing_ index_details
included_ columns	Daftar kolom yang dipisahkan koma, yang diperlukan sebagai kolom untuk *query*.	nvarchar (4000)	sys.dm_db_ missing_ index_details

4.2.1.3. CPU Utilization

CPU Utilization merupakan metrik yang menampilkan informasi tentang historis penggunaan CPU pada SQL Server dan juga dapat mendeteksi hal apa yang menyebabkan penggunaan CPU yang tinggi, misalnya query yang banyak pada satu transaksi. Konsumsi sumber daya CPU yang besar dapat menyebabkan kinerja server menjadi lambat dan tidak optimal.

Stored procedure yang digunakan untuk mengambil data-data yang memuat informasi tentang kondisi CPU terkini pada server yaitu **MS_PerfDashboard.usp_Main_GetCPUHistory** Berdasarkan query yang digunakan, stored procedure ini mengambil data dari DMV diantaranya **sys.dm_os_sys_info**, dan **sys.dm_os_ring_buffers.**

- ***sys.dm_os_sys_info*** merupakan DMV yang mencatat sekumpulan informasi berguna tentang komputer, dan tentang sumber daya yang tersedia untuk digunakan oleh SQL Server [26].

- ***sys.dm_os_ring_buffers*** merupakan catatan peringatan di dalam sistem. Ada sejumlah alert berbeda yang bisa dilihat melalui DMV ini. DMV ini tidak terdapat pada dokumentasi online Microsoft SQL Server, karena DMV ini hanya digunakan untuk diagnostik internal yang suatu saat dapat berubah.

Query yang digunakan pada *stored procedure* **MS_PerfDashboard.usp_Main_GetCPUHistory** adalah sebagai berikut:

```
1.  BEGIN
2.  DECLARE @ms_now bigint
3.  SELECT @ms_now = ms_ticks FROM sys.dm_os_sys_info;

4.  SELECT top 15 record_id,
5.         dateadd(ms, -
    1 * (@ms_now - [timestamp]), GetDate()) AS EventTim
    e, SQLProcessUtilization, SystemIdle,100 - SystemId
    le - SQLProcessUtilization AS OtherProcessUtilizati
    on
6.  FROM (
7.     SELECT record.value('(./Record/@id)[1]', 'int')
    AS record_id,
8.         record.value('(./Record/SchedulerMonitorEvent
    /SystemHealth/SystemIdle)[1]', 'int') AS SystemIdle
    ,
9.         record.value('(./Record/SchedulerMonitorEvent
    /SystemHealth/ProcessUtilization)[1]', 'int') AS SQ
    LProcessUtilization,
10.        timestamp
11.    FROM (
12.        SELECT timestamp, convert(xml, record) AS rec
    ord
13.        FROM sys.dm_os_ring_buffers
```

```
14.        WHERE ring_buffer_type = N'RING_BUFFER_SCHEDU
    LER_MONITOR'
15.        AND record LIKE '%<SystemHealth>%') AS x
16.     ) AS y
17.     ORDER BY record_id DESC
18. END
```

Gambar 4.3 Query pada *stored procedure* MS_PerfDashboard.usp_Main_GetCPUHistory

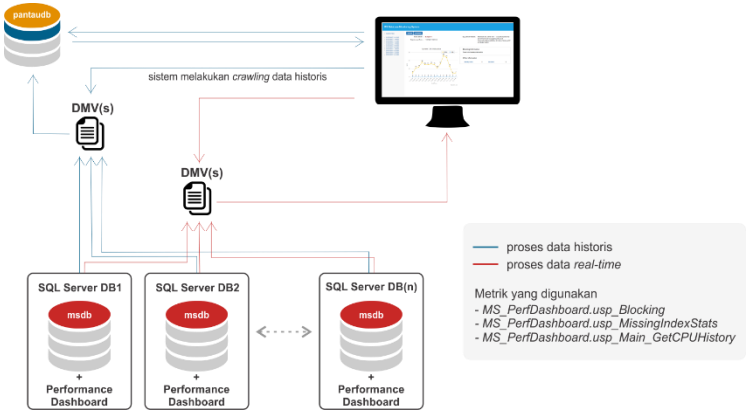
Jika *query* tersebut dijalankan, maka akan menghasilkan tabel dengan kolom-kolom yang memuat informasi tentang keadaan CPU server basis data seperti pada Tabel 4.4

Tabel 4.4 Daftar kolom pada *MS_PerfDashboard.usp_Main_GetCPUHistory* yang digunakan

Nama Kolom	Deskripsi	Tipe Data	Sumber DMV
event_time	Waktu kinerja CPU pada saat tertentu.	timestamp	sys.dm_os_ring_buffers
other_process_utilization	Jumlah persentase kinerja CPU yang digunakan oleh proses lain.	int	sys.dm_os_ring_buffers
sql_process_utilization	Jumlah persentase kinerja CPU yang digunakan oleh proses SQL/*Query*.	int	sys.dm_os_ring_buffers

4.2.2. Desain Arsitektur Sistem

Desain Arsitektur Sistem yang akan digunakan untuk membuat tools pemantauan kinerja basis data berbasis web menggambarkan cara kerja sistem dan alur informasi pada sistem, yakni sebagai berikut:



Gambar 4.4 Desain Arsitektur Tools Pemantauan Kinerja Basis Data Berbasis Web

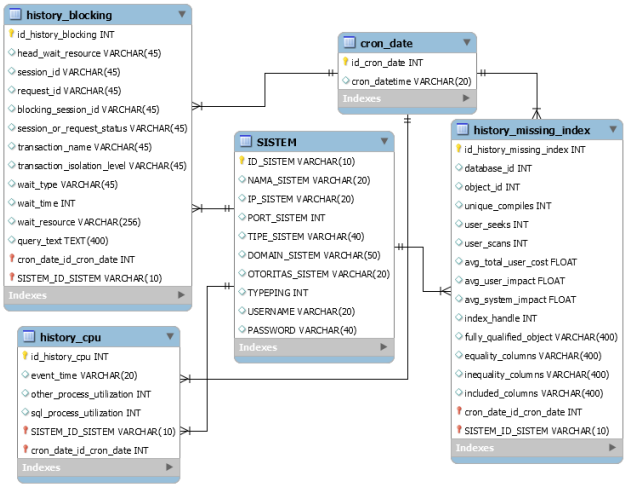
Terdapat dua proses pada tools pemantauan kinerja basis data berbasis web yang akan dibangun, namun terlebih dahulu masing-masing server basis data harus sudah terinstal SQL Server Performance Dashboard agar sistem dapat dijalankan.

Proses yang pertama adalah sistem akan menampilkan informasi kinerja server basis data berdasarkan metrik-metrik yang digunakan secara *real-time*. Cara sistem ini bekerja yaitu dengan memanggil dan menjalankan masing-masing *stored procedure* pada server basis data (basis data yang digunakan adalah Microsoft SQL Server). Data yang diperoleh dari masing-masing *stored procedure* akan ditampilkan secara visual pada sistem web.

Proses yang kedua adalah sistem akan melakukan *crawling* pada server basis data yang sama untuk mendapatkan data historis kinerja server pada waktu-waktu tertentu. Proses *crawling* akan berjalan dengan jarak waktu 10 menit untuk satu kali proses. Data-data historis tersebut akan disimpan pada basis data situs pantau.its.ac.id (basis data yang digunakan adalah MySQL), agar kemudian dapat digunakan oleh admin untuk mengetahui informasi kinerja basis data pada waktu tertentu sesuai waktu yang diinginkan.

4.2.3. Desain Basis Data

Basis data pada pembuatan sistem ini merupakan penambahan dari basis data sistem pantau.its.ac.id yang telah ada sebelumnya. Gambar 4.5 menunjukan tabel-tabel beserta relasinya. Terdapat 4 tabel yang ditambahkan untuk kebutuhan penyimpanan data historis yang merupakan hasil proses *crawling* dari sistem yang dibuat ke server basis data yaitu tabel *history_blocking*, *history_cpu*, *history_missing_index*, dan *cron_date*.



Gambar 4.5 Skema basis data untuk Tools Pemantauan Kinerja Basis Data Berbasis Web

Berikut merupakan penjelasan masing-masing entitas dan atribut-atribut yang terdapat pada masing-masing entitas.

Tabel 4.5 Penjelasan entitas basis data

No.	Tabel	Atribut	Keterangan
1.	cron_date	• id_cron • cron_datetime	Tabel ini digunakan untuk menyimpan waktu ketika proses *crawling* dijalankan
2.	history_ blocking	• id_history_blocking • head_wait_resource • session_id • request_id • blocking_session_id • session_or_ request_status • transaction_name • transaction_ isolation_ level • wait_type • wait_time • wait_resource • query_text	Tabel ini digunakan untuk menyimpan data historis ketika terdapat *blocking* pada waktu tertentu

3.	history_ missing_ index	• id_history_missing_ index • database_id • object_id • unique_compiles • user_seeks • user_scans • avg_total_user_cost • avg_user_impact • avg_system_impact • index_handle • fully_qualified_ object • equality_columns • inequality_columns • included_columns	Tabel ini digunakan untuk menyimpan data historis ketika terdeteksi adanya *missing index* pada waktu tertentu
4.	history_cpu	• id_history_cpu • event_time • other_process_ utilization • sql_process_ utilization	Tabel ini digunakan untuk menyimpan data historis kinerja CPU pada waktu tertentu

Pada skema basis data terdapat beberapa *foreign key* dengan relasi *one-to-many* antar tabel. Berikut adalah penjelasan masing-masing relasi:

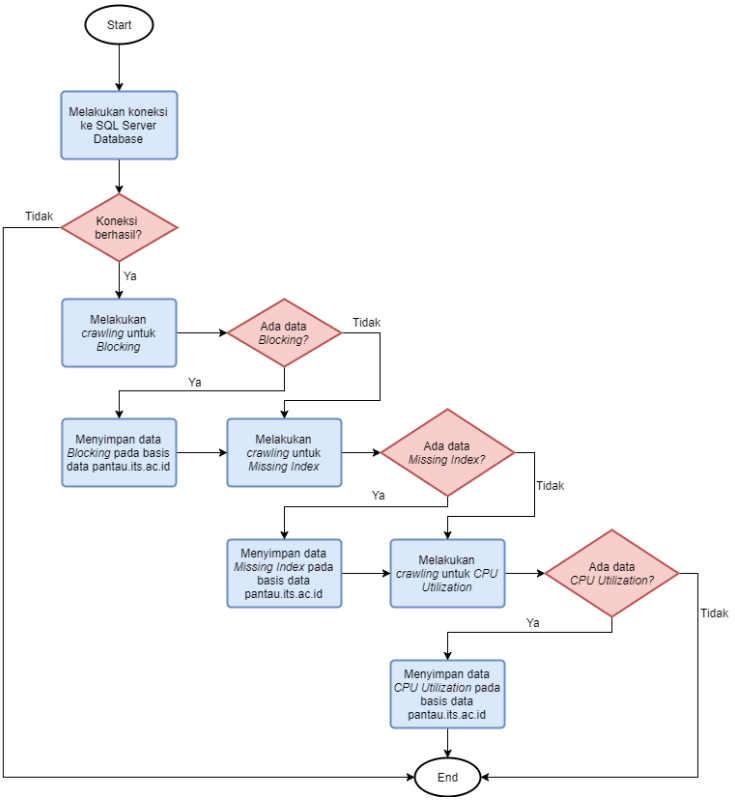
- *fk_history_blocking_cron_date*, merupakan relasi antar entitas *history_blocking* dengan *cron_date* dimana di dalam beberapa *history_blocking* terdapat satu *cron_date* dan sebaliknya.
- *fk_history_blocking_SISTEM*, merupakan relasi antar entitas *history_blocking* dengan *SISTEM* dimana di

dalam beberapa *history_blocking* terdapat satu *SISTEM* dan sebaliknya.

- *fk_history_missing_index_cron_date*, merupakan relasi antar entitas *history_missing_index* dengan *cron_date* dimana di dalam beberapa *history_missing_index* terdapat satu *cron_date* dan sebaliknya.
- *fk_history_missing_index_SISTEM*, merupakan relasi antar entitas *history_missing_index* dengan *SISTEM* dimana di dalam beberapa *history_missing_index* terdapat satu *SISTEM* dan sebaliknya.
- *fk_history_cpu_cron_date*, merupakan relasi antar entitas *history_cpu* dengan *cron_date* dimana di dalam beberapa *history_cpu* terdapat satu *cron_date* dan sebaliknya.
- *fk_history_cpu_SISTEM*, merupakan relasi antar entitas *history_cpu* dengan *SISTEM* dimana di dalam beberapa *history_cpu* terdapat satu *SISTEM* dan sebaliknya.

4.2.4. Desain Crawler

Proses *crawling* pada penelitian ini dibuat dan dirancang dengan menggunakan bahasa pemrograman PHP. Proses ini dilakukan untuk mengambil data historis terkait kondisi server basis data pada waktu-waktu tertentu. Secara garis besar, alur proses *crawling* sistem dijelaskan pada gambar 4.6 sebagai berikut:



Gambar 4.6 *Flowchart* proses *crawling* pada sistem

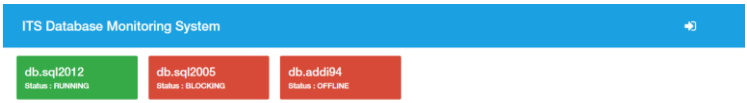
1. *Crawler* akan melakukan koneksi ke server basis data yang telah terinstal SQL Server Performance Dashboard. Apabila koneksi tidak berhasil, maka proses *crawling* akan dihentikan.
2. Setelah koneksi berhasil dilakukan, *crawler* akan mengambil data tentang *Blocking* dengan mengeksekusi *stored procedure* yang ada. Apabila terdapat *blocking* pada server, *crawler* akan menyimpan data tersebut beserta data waktu proses *crawling* dilakukan ke basis data sistem pantau.its.ac.id.

3. Apabila penyimpanan data *blocking* telah selesai atau tidak ada data tentang *blocking* pada server, selanjutnya *crawler* akan mengambil data tentang *Missing Index* dengan mengeksekusi *stored procedure* yang ada. Apabila *missing index* terdeteksi pada server, *crawler* akan menyimpan data tersebut beserta data waktu proses *crawling* dilakukan ke basis data sistem pantau.its.ac.id.
4. Apabila penyimpanan data *missing index* telah selesai atau tidak ada *missing index* yang terdeteksi pada server, selanjutnya *crawler* akan mengambil data *CPU Utilization* dengan mengeksekusi *stored procedure* yang ada. Selanjutnya *crawler* akan menyimpan data tersebut beserta data waktu proses *crawling* dilakukan ke basis data sistem pantau.its.ac.id dan proses *crawling* telah selesai.

4.2.5. Desain Antarmuka Sistem

Desain antarmuka sistem menunjukan bagaimana tampilan sistem yang digunakan untuk berinteraksi dengan pengguna.. Berikut adalah beberapa tampilan antarmuka sistem yang akan dibuat pada penelitian ini berdasarkan kebutuhan fungsional yang telah dijelaskan pada bagian sebelumnya:

4.2.5.1. Halaman kondisi server basis data saat ini.

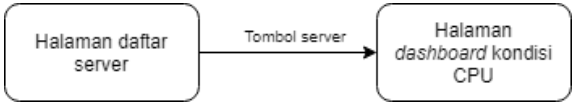


Gambar 4.7 Tampilan antarmuka kondisi server basis data

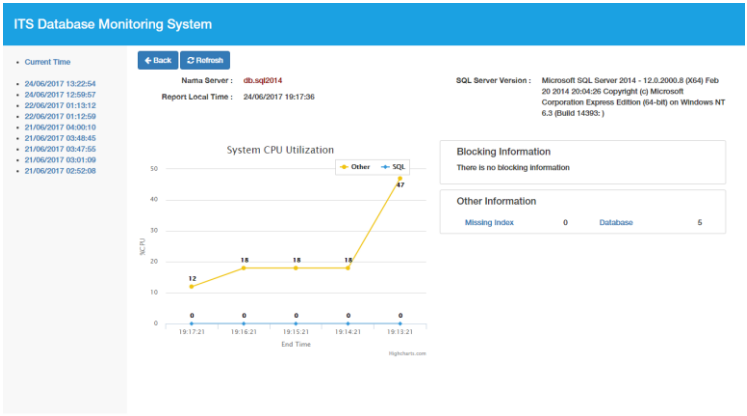
Pada Gambar 4.7, sistem akan menampilkan kondisi server basis data saat ini. Sistem akan menampilkan warna hijau jika

server sedang berjalan dengan baik dan akan menampilkan warna merah jika server sedang dalam keadaan *offline* atau sistem gagal melakukan koneksi ke server tersebut. Selain itu, sistem juga akan menampilkan warna merah jika terjadi *Blocking* pada server yang menandakan bahwa server harus segera ditindaklanjuti agar kinerja server kembali berjalan dengan baik.

4.2.5.2. Halaman informasi penggunaan CPU saat ini.



Gambar 4.8 *UI Storyboard* antarmuka informasi penggunaan CPU saat ini

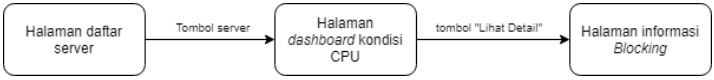


Gambar 4.9 Tampilan informasi tentang detail dan penggunaan CPU

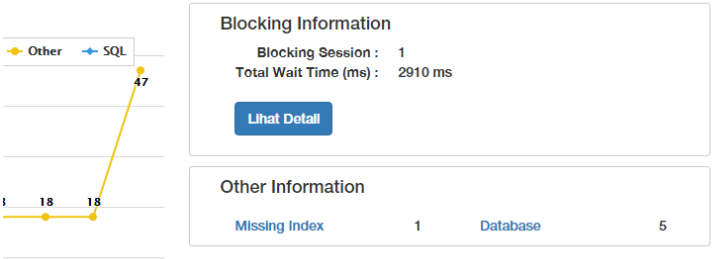
Berdasarkan Gambar 4.8, halaman informasi tentang kondisi CPU saat ini dapat dilihat dengan melakukan klik pada server yang diinginkan. Tombol klik tersebut akan berfungsi jika server sedang tidak dalam keadaan *offline*. Selanjutnya sistem akan menampilkan *dashboard* yang berisi keterangan detail

tentang CPU dan chart kondisi keadaan CPU dalam 15 menit terakhir saat server bekerja, serta menampilkan menu untuk melihat kondisi CPU berdasarkan metrik yang digunakan. Pada sisi kiri terdapat menu untuk melihat data historis CPU berdasarkan waktu yang ada.

4.2.5.3. Halaman informasi adanya *Blocking.*

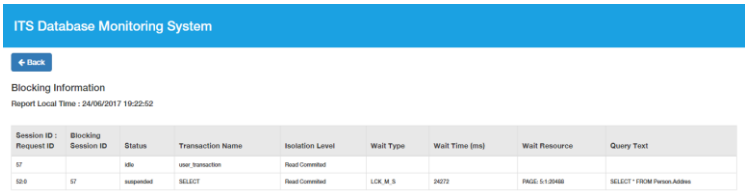


Gambar 4.10 *UI Storyboard* antarmuka informasi adanya *Blocking*



Gambar 4.11 Tampilan informasi *Blocking* pada *dashboard*

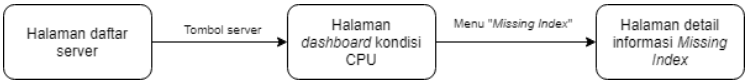
Gambar 4.11 menunjukan tampilan *dashboard* jika terjadi *Blocking* pada server. *Dashboard* akan menampilkan jumlah session/user yang sedang menunggu suatu transaksi akan selesai dan lama waktu session menunggu. Untuk melihat detail informasi *Blocking*, admin dapat melakukan klik pada tombol "Lihat Detail", kemudian sistem akan menampilkan halaman detail informasi *Blocking*.



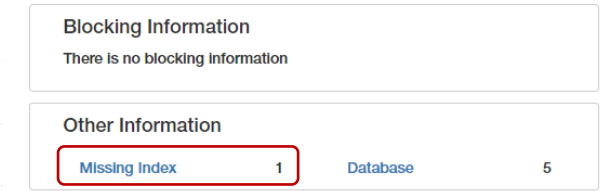
Gambar 4.12 Tampilan detail informasi jika terjadi *Blocking*

Gambar 4.12 menunjukan tampilan antarmuka detail informasi jika terjadi *Blocking* pada server. Sistem akan menampilkan detail informasi berdasarkan data yang didapatkan saat memanggil *stored procedure* pada server basis data.

4.2.5.4. Halaman informasi saat *Missing Index* terdeteksi.



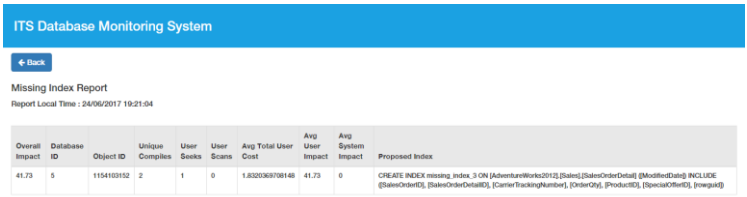
Gambar 4.13 *UI Storyboard* antarmuka informasi detail *Missing Index*



Gambar 4.14 Tampilan informasi *Missing Index* pada *dashboard*

Gambar 4.14 menunjukan tampilan pada *dashboard* jika terdeteksi *Missing Index* pada server. Sistem akan menampilkan jumlah *Missing Index* yang terdeteksi pada server. Untuk melihat detail informasinya, administrator dapat melakukan klik pada *link* menu “Missing Index”. Kemudian sistem akan

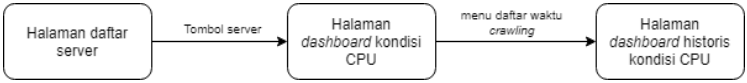
menuju halaman detail informasi *Missing Index* seperti yang ditunjukkan pada gambar 4.15 berikut.



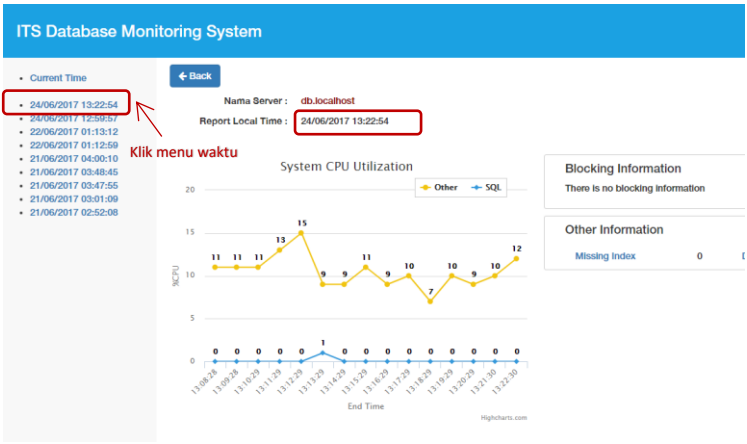
Gambar 4.15 Halaman informasi tentang *missing index* yang terdeteksi pada server

Gambar 4.15 menunjukan tampilan antarmuka detail informasi jika terdapat *missing index* yang terdeteksi pada server. Sistem akan menampilkan detail informasi berdasarkan data yang didapatkan saat memanggil *stored procedure* pada server basis data.

4.2.5.5. Halaman data historis kondisi CPU pada waktu tertentu.



Gambar 4.16 *UI Storyboard* antarmuka informasi historis penggunaan CPU



Gambar 4.17 Tampilan *dashboard* informasi tentang historis penggunaan CPU pada waktu tertentu

Untuk melihat informasi historis penggunaan CPU pada server, admin dapat melakukan klik pada menu waktu yang tersedia pada sisi kanan tampilan sistem. Kemudian sistem akan menunjukkan historis penggunaan CPU sesuai dengan waktu yang dipilih oleh admin.

BAB V
IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai implementasi terhadap perancangan sistem yang telah dilakukan pada bab sebelumnya. Pada bagian implementasi ini akan dijelaskan mengenai lingkungan implementasi, pembuatan fungsi-fungsi sistem dalam bentuk kode program, serta proses pengujian sistem yang akan dilakukan.

5.1. Lingkungan Implementasi

Sistem tools pemantauan kinerja basis data berbasis web berjalan pada server *localhost* sebelum diimplementasikan pada situs pantau.its.ac.id. Spesifikasi komputer yang digunakan pada pembuatan sistem ini adalah sebagai berikut:

Tabel 5.1 Spesifikasi komputer server *localhost*

Processor	Intel(R) Core(TM) i5-2450M CPU @ 2.50GHz
Memory	6,00 GB
Operating System	Windows 10 Pro (x64)

Selain itu, dalam pembuatan sistem ini juga menggunakan beberapa teknologi yang mendukung seperti *text editor, DBMS,* bahasa pemrograman, dan *library* seperti pada tabel 5.2 berikut.

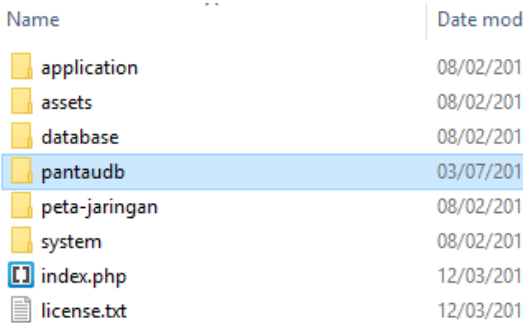
Tabel 5.2 Teknologi yang digunakan

Webserver	Apache 2.4.16
Bahasa Pemrograman	PHP 5.6.12, Javascript
DBMS	MySQL 5.6.26
Text Editor	Brackets 1.8

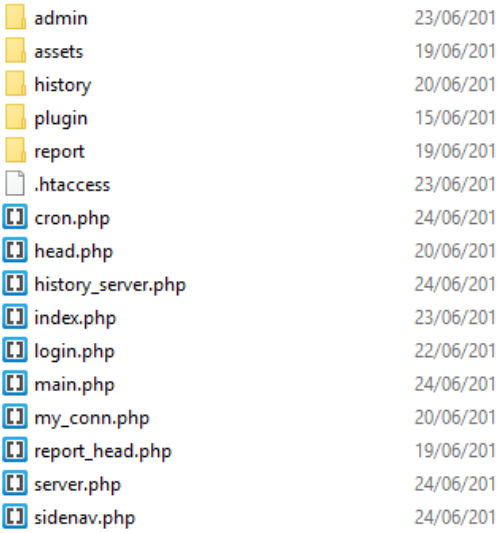
Web Browser	Google Chrome Version 59.0.3071.115 (Official Build) (64-bit)
Library	• JQuery v1.12.4 (Javascript library) • Bootstrap v3.3.7 (CSS Framework & Grid System) • Font Awesome v4.7.0 (Icon) • Highchart v4.2.1 (Chart)

5.2. Direktori Aplikasi

Direktori sistem pemantauan kinerja basis data berbasis web yang dikembangkan ditempatkan sebagai subdirektori situs pantau.its.ac.id



Gambar 5.1 Direktori situs pantau.its.ac.id beserta sub-direktorinya



Gambar 5.2 Direktori tools pemantauan kinerja basis data berbasis web

5.3. Konfigurasi Aplikasi

Sebelum mengembangkan fungsi utama sistem, pertama-tama koneksi ke server DMBS harus dibuat. File my_conn.php terdapat pada direktori utama sistem. Konfigurasi yang dilakukan adalah mengatur nama basis data, *username*, dan *password* seperti pada Gambar 5.3 berikut.

```
1.  <?php
2.    $db = mysqli_connect('localhost', 'root', '',
    'kp_monitoring') or die (koneksi_gagal());
3.
4.    function koneksi_gagal(){
5.      echo "Koneksi tidak berhasil";
6.    }
7.  ?>
```

Gambar 5.3 Potongan kode untuk melakukan koneksi ke basis data

5.4. Pembuatan Aplikasi

Pada bagian ini akan dijelaskan implementasi desain sistem pada bagian sebelumnya ke dalam bahasa pemrograman PHP dan Javascript.

5.4.1. Fungsi *crawler*

Fungsi *crawler* berjalan pada background server, dan dipanggil setiap 10 menit sekali untuk mengambil data historis pada masing-masing server basis data yang telah terinstal SQL Performance Dashboard. Berikut adalah potongan kode program fungsi *crawler*.

```
1.  <?php
2.    include('my_conn.php');
3.    $sql = "SELECT * FROM sistem";
4.    $res = mysqli_query($db, $sql) or die(mysqli_erro
    r($db));
5.
6.    date_default_timezone_set("Asia/Jakarta");
7.    $date = date("d/m/Y H:i:s");
8.    $query = "INSERT INTO cron_date(cron_datetime)
    VALUES ('$date')";
9.    mysqli_query($db, $query);
10.
11.   while($server = mysqli_fetch_array($res)){
12.     $id = $server['ID_SISTEM'];
13.     $sql2 = mysqli_query($db, "SELECT * FROM sistem
     WHERE TIPE_SISTEM = 'DATABASE' AND ID_SISTEM = '$i
    d'") or die(mysqli_error($db));
14.     $ip = mysqli_fetch_array($sql2);
15.
16.     $sql3 = mysqli_query($db, "SELECT id_cron FROM
    cron_date WHERE cron_datetime = '$date'") or die(my
    sqli_error($db));
17.     $cron = mysqli_fetch_array($sql3);
18.
19.     //KONEKSI KE SQL SERVER
20.     $serverName = $ip['IP_SISTEM'];
21.     $uid        = $ip['USERNAME'];
22.     $pwd        = $ip['PASSWORD'];
```

```
23.     $connectionInfo = array( "Database"=>"msdb", "U
    ID"=>$uid, "PWD"=>$pwd);
24.     $db2 = sqlsrv_connect($serverName, $connectionI
    nfo) or die( print_r( sqlsrv_errors(), true));
```

Gambar 5.4 Potongan kode cron.php untuk melakukan koneksi ke masing-masing server basis data

Pertama-tama program akan mengambil data server basis data yang terdapat pada basis data situs pantau.its.ac.id untuk melakukan koneksi ke tiap server. Kemudian program akan mengatur waktu saat fungsi *crawling* dijalankan. Waktu proses *crawling* berjalan ini nantinya akan disimpan pada table cron_date di database situs pantau.its.ac.id. Selanjutnya program akan melakukan koneksi ke masing-masing server dengan username dan password yang ada. Jika koneksi gagal, maka program akan menampilkan error.

```
1.  $cpu01 = $cron['id_cron'];
2.  $cpu02 = $id;
3.  $sql4 = sqlsrv_query($db2, "EXEC MS_PerfDashboard.u
    sp_Main_GetCPUHistory");
4.
5.  while ($cpu = sqlsrv_fetch_array($sql4, SQLSRV_FETC
    H_ASSOC)){
6.    $cpu03 = $cpu['EventTime']->format('H:i:s');
7.    $cpu04 = $cpu['OtherProcessUtilization'];
8.    $cpu05 = $cpu['SQLProcessUtilization'];
9.
10.   mysqli_query($db, "INSERT INTO history_cpu (cron_
    date, nama_sistem, event_time,other_process_utiliza
    tion, sql_process_utilization)
11.   VALUES ('$cpu01','$cpu02','$cpu03','$cpu04','$cpu
    05')");
12. }
```

Gambar 5.5 Potongan cron.php untuk melakukan *crawling* CPU Utilization

Selanjutnya program *crawling* akan berjalan dengan melakukan proses *crawling* untuk metrik CPU Utilization. Program akan memanggil *stored procedure* MS_PerfDashboard.usp_Main_

GetCPUHistory. Data-data yang telah diambil disimpan pada table history_cpu.

```
1.  $blck01 = $cron['id_cron'];
2.  $blck02 = $id;
3.
4.  $sql5 = sqlsrv_query($db2, "EXEC MS_PerfDashboard.u
    sp_Blocking");
5.  while ($block = sqlsrv_fetch_array($sql5, SQLSRV_FE
    TCH_ASSOC)){
6.    if(is_null($block)){
7.      mysqli_query($db, "INSERT INTO history_blocking
     (cron_date, nama_sistem)
8.      VALUES ('$blck01','$blck02')");
9.    } else {
10.    $blck03 = $block['head_wait_resource'];
11.    $blck04 = $block['session_id'];
12.    $blck05 = $block['request_id'];
13.    $blck06 = $block['blocking_session_id'];
14.    $blck07 = $block['session_or_request_status'];
15.    $blck08 = $block['transaction_name'];
16.    $blck09 = $block['transaction_isolation_level'];
17.    $blck10 = $block['wait_type'];
18.    $blck11 = $block['wait_time'];
19.    $blck12 = $block['wait_resource'];
20.    $blck13 = $block['query_text'];
21.
22.     mysqli_query($db, "INSERT INTO history_blocking
     (cron_date, nama_sistem, head_wait_resource, sessi
    on_id, request_id, blocking_session_id, session_or_
    request_status, transaction_name, transaction_isola
    tion_level, wait_type, wait_time, wait_resource, qu
    ery_text)
23. VALUES('$blck01','$blck02','$blck03','$blck04','$bl
    ck05','$blck06','$blck07','$blck08','$blck09','$blc
    k10','$blck11','$blck12','$blck13')");
24.   }  }
```

Gambar 5.6 Potongan cron.php untuk melakukan *crawling* Blocking

Setelah proses *crawling* CPU Utilization selesai, selanjutnya program akan melakukan proses *crawling* untuk Blocking. Program akan memanggil *stored procedure*

MS_PerfDashboard.usp_Blocking. Apabila terdapat *blocking* pada server basis data, maka data-data yang diambil disimpan ke dalam table history_blocking. Jika tidak terdapat *blocking* pada server, maka program akan melanjutkan proses berikutnya.

```
1.  $mi01 = $cron['id_cron'];
2.  $mi02 = $id;
3.  $sql6 = sqlsrv_query($db2, "EXEC MS_PerfDashboard.u
    sp_MissingIndexStats @DatabaseID = NULL, @ObjectID
    = NULL");
4.  while ($miss_index = sqlsrv_fetch_array($sql6, SQLS
    RV_FETCH_ASSOC)){
5.    $mi04 = $miss_index['database_id'];
6.    $mi05 = $miss_index['object_id'];
7.    $mi06 = $miss_index['unique_compiles'];
8.    $mi07 = $miss_index['user_seeks'];
9.    $mi08 = $miss_index['user_scans'];
10.   $mi09 = $miss_index['avg_total_user_cost'];
11.   $mi10 = $miss_index['avg_user_impact'];
12.   $mi11 = $miss_index['avg_system_impact'];
13.   $mi12 = $miss_index['index_handle'];
14.   $mi13 = $miss_index['fully_qualified_object'];
15.   $mi14 = $miss_index['equality_columns'];
16.   $mi15 = $miss_index['inequality_columns'];
17.   $mi16 = $miss_index['included_columns'];
18.
19.   mysqli_query($db, "INSERT INTO history_missing_in
    dex (cron_date, nama_sistem, database_id, object_id
    , unique_compiles, user_seeks, user_scans, avg_tota
    l_user_cost, avg_user_impact, avg_system_impact, in
    dex_handle, fully_qualified_object, equality_column
    s, inequality_columns, included_columns)
20. VALUES('$mi01','$mi02','$mi04','$mi05','$mi06','$mi
    07','$mi08','$mi09','$mi10','$mi11','$mi12','$mi13'
    ,'$mi14','$mi15','$mi16')"); }
```

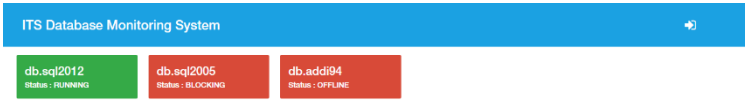
Gambar 5.7 Potongan cron.php untuk melakukan *crawling* Missing Index

Proses terakhir dalam fungsi *crawling* adalah melakukan *crawling* untuk *missing index.* Program akan memanggil *stored procedure* MS_PerfDashboard.usp_MissingIndexStats.

Apabila *missing index* terdeteksi pada server basis data, maka data-data yang diambil disimpan ke dalam table history_missing_index. Jika *missing index* tidak terdeteksi pada server, maka program akan menghentikan proses *crawling*.

5.4.2. Fungsi menampilkan kondisi server basis data

Pada saat admin menjalankan sistem, tampilan pertama yang akan muncul adalah antarmuka daftar server seperti pada Gambar 5.8. Pada tampilan akan terdapat informasi apakah server sedang dalam keadaan normal atau sedang *offline*, maupun ketika terdapat *blocking* pada server.



Gambar 5.8 Tampilan antarmuka kondisi server basis data saat ini

Untuk mendapatkan informasi server saat ini, terlebih dahulu dilakukan koneksi ke server tersebut. Gambar 5.9 adalah potongan kode program untuk melakukan koneksi ke masing-masing server. Jika program gagal melakukan koneksi, maka akan muncul keterangan bahwa server sedang dalam keadaan *offline*.

```
1.  <?php
2.    $sql = "SELECT * FROM sistem";
3.    $res = mysqli_query($db, $sql) or die(mysqli_erro
    r($db));
4.
5.    while($row = mysqli_fetch_array($res)){
6.      $serverName = $row['IP_SISTEM'];
7.      $uid        = $row['USERNAME'];
8.      $pwd        = $row['PASSWORD'];
9.      $connectionInfo = array( "Database"=>"msdb", "U
    ID"=>$uid, "PWD"=>$pwd);
```

```
10.     $db2 = sqlsrv_connect( $serverName, $connection
    Info);
11.
12.     if($db2 === false){
13. ?>
14.     <div class="col-md-2 nama-sistem-merah">
15.       <div class="col-md-12" style="padding: 0;">
16.         <a href="#"><?php echo $row['NAMA_SISTEM'];
    ?></a>
17.       </div>
18.       <div class="col-md-
    12"  style="padding: 0;">
19.         <p style="color:#fff;">Status : <?php echo
    "OFFLINE";?></p>
20.       </div>
21.     </div>
```

Gambar 5.9 Potongan main.php untuk melakukan koneksi ke server basis data

Jika program berhasil melakukan koneksi ke setiap server, selanjutnya program akan mendeteksi apakah sedang terjadi *blocking* pada server atau tidak. Gambar 5.10 menunjukan potongan program untuk mendeteksi *blocking* pada server. Jika tidak terdapat *blocking*, maka akan muncul keterangan bahwa server berjalan dengan baik.

```
1.  <?php
2.    } else {
3.      $sql2 = "EXEC MS_PerfDashboard.usp_Blocking";
4.      $res2 = sqlsrv_query($db2, $sql2);
5.      $block = sqlsrv_fetch_array($res2, SQLSRV_FETCH
    _ASSOC);
6.
7.      if (is_null($block['session_id'])){
8.  ?>
9.      <div class="col-md-2 nama-sistem-hijau">
10.       <div class="col-md-12" style="padding: 0;">
11.         <a href="server.php?id=<?php echo $row['ID_
    SISTEM']; ?>"><?php echo $row['NAMA_SISTEM'];?></a>

12.       </div>
```

```
13.       <div class="col-md-
    12"  style="padding: 0;">
14.         <p style="color:#fff;">Status : <?php echo
    "RUNNING";?></p>
15.       </div>
16.     </div>
```

Gambar 5.10 Potongan main.php untuk mendeteksi *blocking* pada server basis data

Namun, jika terdapat *blocking* pada server, maka sistem akan memunculkan keterangan bahwa sedang terdapat *blocking* pada server saat ini.

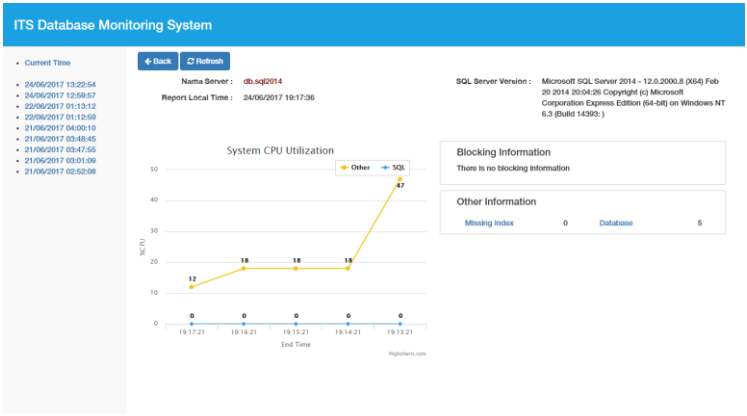
```
1.  <?php
2.    } else {
3.  ?>
4.    <div class="col-md-2 nama-sistem-merah">
5.      <div class="col-md-12" style="padding: 0;">
6.        <a href="server.php?id=<?php echo $row['ID_SI
    STEM']; ?>"><?php echo $row['NAMA_SISTEM'];?></a>
7.      </div>
8.      <div class="col-md-12" style="padding: 0;">
9.        <p style="color:#fff;">Status : <?php echo "B
    LOCKING";?></p>
10.     </div>
11.   </div>
```

Gambar 5.11 Potongan main.php jika terdeteksi *blocking* pada server basis data

5.4.3. Fungsi menampilkan informasi *CPU Utilization*

Untuk menuju ke halaman *dashboard* server, terlebih dahulu admin harus melakukan klik pada server yang ingin dipantau. Selanjutnya akan muncul tampilan antarmuka seperti pada Gambar 5.12. Informasi mengenai kondisi CPU ditampilkan dalam bentuk line chart untuk mengetahui trend persentase penggunaan CPU pada server. Terdapat dua informasi pada

dashboard yaitu penggunaan CPU karena proses SQL dan penggunaan CPU karena proses yang lain.



Gambar 5.12 Tampilan antarmuka *dashboard*

Untuk menampilkan informasi kondisi CPU, terlebih dahulu program akan melakukan koneksi ke server yang dipilih oleh administrator sebelumnya. Gambar 5.13 menunjukkan potongan kode program untuk melakukan koneksi dengan menggunakan ID server sebagai *identifier*. Jika program gagal melakukan koneksi, maka akan muncul pesan error yang dibuat pada fungsi koneksi_error().

```
1.  include('my_conn.php');
2.
3.  $id = $_GET['id'];
4.  $sql= "SELECT * FROM sistem WHERE ID_SISTEM = '$id'
    ;";
5.  $res = mysqli_query($db, $sql) or die(mysqli_error
    ($db));
6.  $sistem = mysqli_fetch_array($res);
7.
8.  //KONEKSI KE SQL SERVER
9.  $serverName = $sistem['IP_SISTEM'];
10. $uid        = $sistem['USERNAME'];
11. $pwd        = $sistem['PASSWORD'];
12. $connectionInfo = array( "Database"=>"msdb", "UID"
    =>$uid, "PWD"=>$pwd);
```

```
13. $db2 = sqlsrv_connect( $serverName, $connectionInfo
    ) or die(koneksi_error());
14.
15. function koneksi_error(){
16.   echo "Koneksi tidak berhasil";
17. }
```

Gambar 5.13 Potongan server.php untuk melakukan koneksi ke server basis data

Setelah koneksi berhasil dilakukan, selanjutnya program akan memanggil *stored procedure* MS_PerfDashboard.usp_Main_ GetCPUHistory yang menyimpan data kondisi CPU saat ini dengan kode program seperti yang ditunjukkan pada Gambar 5.14.

```
1.  <div class="col-md-6" id="cpu" style="padding-
    left:0;">
2.  <?php
3.    $sql = "EXEC MS_PerfDashboard.usp_Main_GetCPUHist
    ory";
4.    $res = sqlsrv_query($db2, $sql);
5.
6.    while ($cpu = sqlsrv_fetch_array($res, SQLSRV_FET
    CH_ASSOC)){
7.      $cpu_time[] = "'".$cpu['EventTime']-
    >format('H:i:s')."'";
8.      $other_time[]= $cpu['OtherProcessUtilization'];
9.      $sql_time[] = $cpu['SQLProcessUtilization'];
10.   }
11. ?>
```

Gambar 5.14 Potongan server.php untuk menampilkan data *CPU Utilization*

Selanjutnya data-data tersebut akan ditampilkan ke dalam bentuk chart. Untuk pembuatan sistem ini, penulis menggunakan *library* Highchart yang merupakan *library* Javascript untuk menampilkan chart pada aplikasi web. Gambar 5.15 merupakan potongan kode Javascript untuk menampilkan informasi kondisi terkini CPU dalam bentuk chart.

```
<script type="text/javascript">
  $(function () {
    $('#cpu').highcharts({
      chart: { type: 'line'},
      title: { text: 'System CPU Utilization' },
      xAxis: {
        title: { text: 'End Time'},
        categories: [<?php echo join($cpu_time, ',' ) ?>]
      },
      yAxis: {
        min: 0,
        title: { text: '%CPU' },
        stackLabels: {
          enabled: false,
          style: {
            fontWeight: 'bold',
            color: (Highcharts.theme && Highcharts.theme.textColor) || 'gray'
          }
        }
      },
      legend: {
        align: 'right',
        x: -30,
        verticalAlign: 'top',
        y: 25,
        floating: true,
        backgroundColor: (Highcharts.theme && Highcharts.theme.background2) || 'white',
        borderColor: '#CCC',
        borderWidth: 1,
        shadow: false
      },
      tooltip: {
        headerFormat: '<b>{point.x}</b><br/>',
        pointFormat: '{series.name}: {point.y}<br/>Total: {point.stackTotal}'
      },
      plotOptions: {
        line: {
          dataLabels: { enabled: true },
          enableMouseTracking: false
        }
      },
```

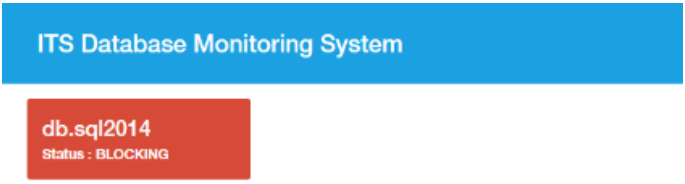
```
42.         series: [{
43.           name: 'Other',
44.           data: [<?php echo join($other_time, ',') ?>
    ],
45.           color: '#f1c40f'
46.         }, {
47.           name: 'SQL',
48.           data: [<?php echo join($sql_time, ',') ?>],

49.           color: '#3498db'
50.         }]
51.       });
52.     });
53. </script>
```

Gambar 5.15 Potongan server.php untuk menampilkan informasi *CPU Utilization* dalam bentuk chart

5.4.4. Fungsi menampilkan informasi *Blocking*

Jika terjadi *blocking* pada server basis data, administrator dapat langsung mengetahuinya pada tampilan antarmuka awal sistem dengan keterangan seperti pada Gambar 5.16



Gambar 5.16 Tampilan antarmuka jika terdeteksi *blocking* pada server

Selanjutnya, untuk mengetahui penyebab terjadinya *blocking* pada server, administrator dapat melakukan klik pada server. Kemudian sistem akan menampilkan antarmuka *dashboard* sistem. Gambar 5.17 menunjukkan tampilan *dashboard* jika terjadi *blocking* pada server basis data.

Gambar 5.17 Tampilan antarmuka pada *dashboard* jika terdeteksi *blocking* pada server

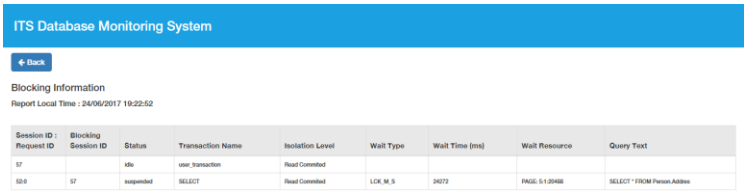
Untuk menampilkan informasi *blocking* pada *dashboard* seperti pada Gambar 5.17, program yang sudah melakukan koneksi ke server basis data selanjutnya memanggil *stored procedure* MS_PerfDashboard.usp_Blocking yang menyimpan data terkait *blocking* pada server seperti pada Gambar 5.18

```
1.  <div class="col-md-
    12 blocking" style="border: 1px solid #cecece; bord
    er-radius:3px;">
2.    <div class="col-md-
    12"><h4>Blocking Information</h4></div>
3.      <div class="col-md-12">
4.  <?php
5.    $wait_time = 0;
6.    $sql2 = "EXEC MS_PerfDashboard.usp_Blocking";
7.    $res2 = sqlsrv_query($db2, $sql2, array(), array(
     "Scrollable" => 'static' ));
8.    $block = sqlsrv_fetch_array($res2, SQLSRV_FETCH_A
    SSOC);
9.
10.   if (is_null($block['head_wait_resource'])){
11.     echo "<p style='margin-
    bottom:20px;'>There is no blocking information";
12.   } else {
13.     $total = 0;
14.     while($data = sqlsrv_fetch_array($res2)) {
15.       $total += $data['wait_time'];
16.     }
17.     $blocking_session = sqlsrv_num_rows($res2)-1;
18.
19.     echo "<dl class='dl-horizontal'>";
20.     echo "<dt>Blocking Session : </dt>";
21.     echo "<dd>".$blocking_session."</dd>";
```

```
22.      echo "<dt>Total Wait Time (ms) : </dt>";
23.      echo "<dd>".$total." ms</dd>";
24.      echo "</dl>";
25.      echo "<div class='col-md-12' style='margin-
    bottom:20px;'><a class ='btn btn-
    primary' href=\"report/blocking_info.php?id={$siste
    m['ID_SISTEM']}\">Lihat Detail</a></div>";
26.  }
27.    ?>
28.    </div>
29.  </div>
```

Gambar 5.18 Potongan server.php untuk menampilkan informasi *blocking* pada antarmuka *dashboard*

Selanjutnya, untuk melihat informasi detail tentang *blocking* pada server basis data, administrator dapat melakukan klik pada tombol “Lihat Detail”. Kemudian sistem akan menampilkan antarmuka detail informasi *blocking* seperti pada gambar 5.19.



Gambar 5.19 Tampilan antarmuka detail informasi *Blocking*

Untuk mendapatkan data tentang detail informasi *blocking* pada server basis data, program menjalankan fungsi get_blocking_information() dimana fungsi ini akan memanggil *stored procedure* untuk *Blocking* dan mengambil data-data yang disimpan seperti pada Gambar 5.20

```
1.   function get_blocking_info($db2){
2.     $sql = "EXEC MS_PerfDashboard.usp_Blocking";
3.     $res = sqlsrv_query($db2, $sql);
4.     while ($block = sqlsrv_fetch_array($res, SQLSRV_F
    ETCH_ASSOC)){
```

```
5.      echo "<tr>";
6.        if (is_null($block['request_id'])){ echo "<td
    >".$block['session_id']."</td>"; }
7.          else { echo "<td>".$block['session_id'].":"
    .$block['request_id']."</td>"; }
8.      echo "<td>".$block['blocking_session_id']."</td
    >";
9.      echo "<td>".$block['session_or_request_status']
    ."</td>";
10.     echo "<td>".$block['transaction_name']."</td>";

11.       if ($block['transaction_isolation_level'] ==
    1){ echo "<td>Read Uncommited</td>"; }
12.         else if ($block['transaction_isolation_leve
    l'] == 2){ echo "<td>Read Commited</td>"; }
13.         else if ($block['transaction_isolation_leve
    l'] == 3){ echo "<td>Repeatable Read</td>"; }
14.         else if ($block['transaction_isolation_leve
    l'] == 4){ echo "<td>Snapshot</td>"; }
15.         else if ($block['transaction_isolation_leve
    l'] == 5){ echo "<td>Serializable</td>"; }
16.     echo "<td>".$block['wait_type']."</td>";
17.     echo "<td>".$block['wait_time']."</td>";
18.     echo "<td>".$block['wait_resource']."</td>";
19.     echo "<td>".$block['query_text']."</td>";
20.     echo "</tr>";
21.   }
22. }
```

Gambar 5.20 Potongan fungsi untuk memanggil *stored procedure* untuk *Blocking*

Selanjutnya, data-data tersebut ditampilkan ke dalam bentuk tabel agar informasi tersebut dapat dengan mudah dimengerti oleh administrator. Gambar 5.21 merupakan potongan kode program untuk menampilkan data ke dalam bentuk tabel.

```
1.  <div class="col-md-12 table-
    responsive" style="margin-top:20px;">
2.    <table class="table table-bordered">
3.      <thead style="font-size:10pt; background-
    color:#e8e8e8;"> <tr>
```

```
4.   <th style="width: 100px;">Session ID : Request ID</
     th>
5.   <th style="width: 100px;">Blocking Session ID</th>
6.           <th>Status</th>
7.           <th>Transaction Name</th>
8.           <th>Isolation Level</th>
9.           <th>Wait Type</th>
10.          <th>Wait Time (ms)</th>
11.          <th>Wait Resource</th>
12.          <th>Query Text</th>
13.        </tr>
14.      </thead>
15.      <tbody style="font-size:10px;">
16.        <?php get_blocking_info($db2); ?> </tbody>
17.    </table>
18. </div>
```

Gambar 5.21 Potongan blocking_info.php untuk menampilkan hasil fungsi dalam bentuk tabel

5.4.5. Fungsi menampilkan informasi *Missing Index*

Jika *missing index* pada suatu basis data terdeteksi oleh sistem, sistem akan menampilkan informasi tersebut pada *dashboard* seperti pada Gambar 5.22.

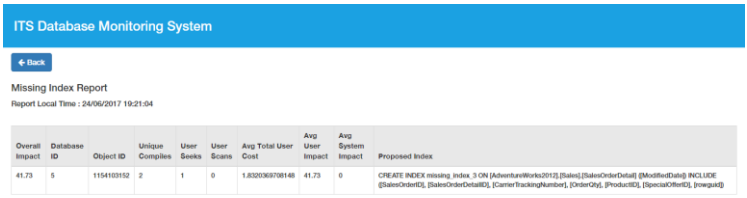
Gambar 5.22 Tampilan antarmuka pada *dashboard* jika terdeteksi *missing index* pada server

Untuk menampilkan informasi *missing index* pada *dashboard* seperti pada Gambar 5.22, program yang sudah melakukan koneksi ke server basis data selanjutnya memanggil *stored procedure* MS_PerfDashboard.usp_MissingIndexStats yang menyimpan data terkait *missing index* pada server seperti pada Gambar 5.23. Pada halaman *dashboard*, akan muncul keterangan jumlah *missing index* yang terdapat pada server.

```
1.   <div class="col-md-6" style="padding-
     top: 10px; padding-bottom:10px;">
2.     <div class="col-md-10">
3.     <?php echo "<a href=\"report/missing_index.php?id
     ={$sistem['ID_SISTEM']}\">Missing Index</a>"?>
4.     </div>
5.     <div class="col-md-2">
6.     <?php
7.       $sql3 = "EXEC MS_PerfDashboard.usp_MissingIndex
     Stats @DatabaseID = NULL, @ObjectID = NULL";
8.       $res3 = sqlsrv_query($db2, $sql3, array(), arra
     y( "Scrollable" => 'static' ));
9.       $row_count = sqlsrv_num_rows($res3);
10.      echo $row_count;
11.    ?>
12.    </div></div>
```

Gambar 5.23 Potongan server.php untuk menampilkan informasi *missing index* pada antarmuka *dashboard*

Selanjutnya, untuk melihat informasi detail tentang *missing index* pada server basis data, administrator dapat melakukan klik menu “Missing Index” pada *dashboard* sistem. Kemudian sistem akan menampilkan antarmuka detail informasi *missing index* seperti pada gambar 5.24.



Gambar 5.24 Tampilan antarmuka detail informasi *Missing Index*

Untuk mendapatkan data tentang detail informasi *missing index* pada server basis data, program menjalankan fungsi get_missing_index_info() dimana fungsi ini akan memanggil

stored procedure untuk *Missing Index* dan mengambil data-data yang disimpan seperti pada Gambar 5.25.

```
1.  function get_missing_index_info($db2){
2.    $sql = "EXEC MS_PerfDashboard.usp_MissingIndexSta
    ts @DatabaseID = NULL, @ObjectID = NULL";
3.    $res = sqlsrv_query($db2, $sql);
4.    while ($miss_index = sqlsrv_fetch_array($res, SQL
    SRV_FETCH_ASSOC)){
5.     $pi01 = $miss_index['index_handle'];
6.     $pi02 = $miss_index['fully_qualified_object'];
7.     $pi03 = $miss_index['equality_columns'];
8.     $pi04 = $miss_index['inequality_columns'];
9.     $pi05 = str_replace(".",",",$miss_index['include
    d_columns']);
10.
11.    echo "<tr>";
12.    echo "<td>".$miss_index['database_id']."</td>";
13.    echo "<td>".$miss_index['object_id']."</td>";
14.    echo "<td>".$miss_index['unique_compiles']."</td
    >";
15.     echo "<td>".$miss_index['user_seeks']."</td>";
16.     echo "<td>".$miss_index['user_scans']."</td>";
17.     echo "<td>".$miss_index['avg_total_user_cost'].
    "</td>";
18.     echo "<td>".$miss_index['avg_user_impact']."</t
    d>";
19.     echo "<td>".$miss_index['avg_system_impact']."<
    /td>";
20.       if(is_null($pi03) && is_null($pi04) && is_nul
    l($pi05)){
21.         echo "<td>CREATE INDEX missing_index_".$pi0
    1." ON ".$pi02."</td>";
22.       } else if (is_null($pi04) && is_null($pi05)){
23.         echo "<td>CREATE INDEX missing_index_".$pi0
    1." ON ".$pi02." (".$pi03.")</td>";
24.       } else if (is_null($pi03) && is_null($pi05)){
25.         echo "<td>CREATE INDEX missing_index_".$pi0
    1." ON ".$pi02." (".$pi04.")</td>";
26.       } else if (is_null($pi05)){
27.         echo "<td>CREATE INDEX missing_index_".$pi0
    1." ON ".$pi02." (".$pi03.",".$pi04.")</td>";
28.       } else if (is_null($pi04)){
```

```
29.         echo "<td>CREATE INDEX missing_index_".$pi0
    1." ON ".$pi02." (".$pi03.") INCLUDE (".$pi05.")</t
    d>";
30.       } else if (is_null($pi03)){
31.         echo "<td>CREATE INDEX missing_index_".$pi0
    1." ON ".$pi02." (".$pi04.") INCLUDE (".$pi05.")</t
    d>";
32.       } else {
33.         echo "<td>CREATE INDEX missing_index_".$pi0
    1." ON ".$pi02." (".$pi03.",".$pi04.") INCLUDE (".$
    pi05.")</td>";
34.       }
35.     echo "</tr>";  } }
```

Gambar 5.25 Potongan fungsi untuk memanggil *stored procedure* untuk *Missing Index*

Selanjutnya, data-data tersebut ditampilkan ke dalam bentuk tabel agar informasi tersebut dapat dengan mudah dimengerti oleh administrator. Gambar 5.26 merupakan potongan kode program untuk menampilkan data ke dalam bentuk tabel.

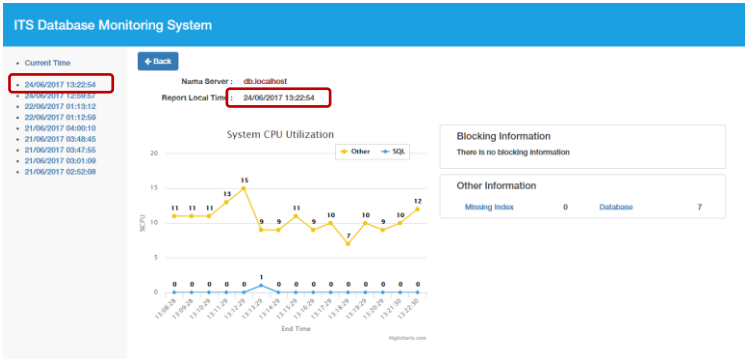
```
1.  <div class="col-md-12 table-
    responsive" style="margin-top:20px;">
2.    <table class="table table-bordered">
3.      <thead style="font-size:10pt; background-
    color:#e8e8e8;">
4.        <tr>
5.          <th>Database ID</th>
6.          <th>Object ID</th>
7.          <th>Unique Compiles</th>
8.          <th>User Seeks</th>
9.          <th>User Scans</th>
10.         <th>Avg Total User Cost</th>
11.         <th>Avg User Impact</th>
12.         <th>Avg System Impact</th>
13.         <th>Proposed Index</th>
14.       </tr>
15.     </thead>
16.     <tbody style="font-size:9pt;">
17.       <?php get_missing_index_info($db2); ?>
18.     </tbody>
```

```
19.    </table>
20.  </div>
```

Gambar 5.26 Potongan missing_index.php untuk menampilkan hasil fungsi dalam bentuk tabel

5.4.6. Fungsi menampilkan informasi historis *CPU Utilization*

Jika admin ingin melihat data historis yang didapatkan melalui proses *crawling*, admin dapat memilih salah satu waktu yang tersedia pada menu sebelah kiri. Kemudian sistem akan menampilkan antarmuka seperti pada Gambar 5.27 berikut.



Gambar 5.27 Tampilan antarmuka *dashboard* historis penggunaan CPU

Sebelum menampilkan data historis pada sistem, terlebih dahulu program melakukan koneksi ke server DBMS tempat menyimpan data-data historis. Potongan kode program pada Gambat 5.28 berikut adalah potongan kode program untuk melakukan koneksi dan mengambil ID sistem serta ID waktu sebagai *identifier* untuk mengambil data historis.

```
1.   $id_time = $_GET['time'];
2.   $id = $_GET['id'];
```

```
3.  $sql = mysqli_query($db, "SELECT * FROM cron_date W
    HERE id_cron = '$id_time'") or die(mysqli_error($db
    ));
4.  $row = mysqli_fetch_array($sql);
5.
6.  $sql2 = mysqli_query($db, "SELECT NAMA_SISTEM FROM
    sistem WHERE ID_SISTEM = '$id'") or die(mysqli_erro
    r($db));
7.  $row2 = mysqli_fetch_array($sql2);
8.
9.  $id_cron = $row['id_cron'];
10. $waktu   = $row['cron_datetime'];
```

Gambar 5.28 Potongan history_server.php untuk melakukan koneksi ke server basis data

Untuk menampilkan data historis CPU, program akan mengambil data historis dari tabel history_cpu dengan ID Sistem dan ID cron_date yang sesuai dengan pilihan admin. Potongan kode program untuk melakukan pengambilan data ditunjukkan pada Gambar 5.29 berikut.

```
1.  <div class="col-md-6" id="cpu2" style="padding-
    left:0;">
2.  <?php
3.    $sql2 = mysqli_query($db, "SELECT * FROM history_
    cpu WHERE cron_date = '$id_time' AND nama_sistem =
    '$id'") or die(mysqli_error($db));
4.    while($cpu = mysqli_fetch_array($sql2)){
5.   $cpu_time[] = "'".$cpu['event_time']."'";
6.   $other_time[] = $cpu['other_process_utilization'];
7.   $sql_time[] = $cpu['sql_process_utilization'];
8.    }
9.  ?>
10. </div>
```

Gambar 5.29 Potongan history_server.php untuk menampilkan data *CPU Utilization*

Setelah data tersebut diambil, selanjutnya program akan menampilkan data tersebut ke dalam bentuk chart dengan menggunakan kode Javascript dari *library* Highchart sama

seperti yang digunakan pada fungsi menampilkan informasi CPU saat ini.

5.4.7. Fungsi menampilkan informasi historis *Blocking*

Secara umum, tampilan antarmuka *dashboard* ketika admin ingin melihat data historis *blocking* tidak jauh berbeda dengan antarmuka *dashboard* ketika terjadi *blocking* pada server basis data secara *real-time* seperti pada Gambar 5.30 berikut ini.

Gambar 5.30 Tampilan antarmuka *dashboard* historis jika terdapat *Blocking*

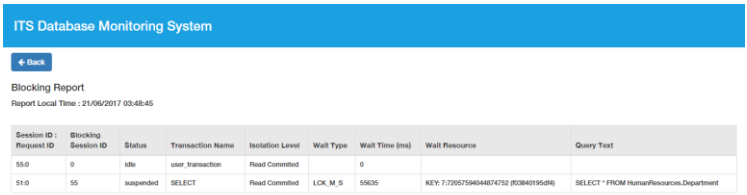
Pertama-tama, setelah program berhasil melakukan koneksi ke server DMBS, program akan mengambil data historis *blocking* dari tabel history_blocking untuk ditampilkan pada antarmuka *dashboard* seperti yang ditunjukkan pada Gambar 5.31 berikut ini.

```
1.   <div class="col-md-
     12 blocking" style="border: 1px solid #cecece; bord
     er-radius:3px;">
2.     <div class="col-md-
     12"><h4>Blocking Information</h4></div>
3.       <div class="col-md-12">
4.       <?php
5.         $wait_time = 0;
```

```
6.        $sql3  = mysqli_query($db, "SELECT * FROM his
    tory_blocking WHERE cron_date = '$id_time' AND nama
    _sistem = '$id'") or die(mysqli_error($db));
7.        $block = mysqli_fetch_array($sql3);
8.        if (is_null($block['head_wait_resource'])){
9.          echo "<p style='margin-
    bottom:20px;'>There is no blocking information";
10.       } else {
11.         $total = 0;
12.         while($data = mysqli_fetch_array($sql3)){
13.           $total += $data['wait_time'];
14.         }
15.     $blocking_session = mysqli_num_rows($sql3)-1;
16.         echo "<dl class='dl-horizontal'>";
17.         echo "<dt>Blocking Session : </dt>";
18.         echo "<dd>".$blocking_session."</dd>";
19.         echo "<dt>Total Wait Time (ms) : </dt>";
20.         echo "<dd>".$total." ms</dd>";
21.         echo "</dl>";
22.         echo "<div class='col-md-12' style='margin-
    bottom:20px;'><a class ='btn btn-
    primary' href=\"history/his_blocking_info.php?time=
    {$id_time}&id={$id}\">Lihat Detail</a></div>";
23.     }
24.   ?>
25.   </div>
26. </div>
```

Gambar 5.31 Potongan history_server.php untuk menampilkan data *Blocking*

Untuk melihat detail informasi historis ketika terjadi *blocking*, administrator dapat melakukan klik pada tombol Lihat Detail. Kemudian sistem akan menampilkan halaman detail informasi *blocking* seperti pada Gambar 5.32 berikut ini.



Gambar 5.32 Tampilan antarmuka detail informasi historis untuk *Blocking*

Untuk melakukan pengambilan data, program akan menjalankan fungsi get_history_blocking_info() yang akan mengambil data dari tabel history_blocking. Gambar 5.33 merupakan potongan kode program untuk memanggil fungsi tersebut.

```
1.  function get_history_blocking_info($db,$id_time,$id
    ){
2.    $sql = "SELECT * FROM history_blocking WHERE cron
    _date = '$id_time' AND nama_sistem = '$id'";
3.    $res = mysqli_query($db, $sql) or die(mysqli_erro
    r($db));
4.    while ($block = mysqli_fetch_array($res)){
5.      echo "<tr>";
6.        if (is_null($block['request_id'])){ echo "<td
    >".$block['session_id']."</td>"; }
7.          else { echo "<td>".$block['session_id'].":"
    .$block['request_id']."</td>"; }
8.      echo "<td>".$block['blocking_session_id']."</td
    >";
9.      echo "<td>".$block['session_or_request_status']
    ."</td>";
10.     echo "<td>".$block['transaction_name']."</td>";
11.       if ($block['transaction_isolation_level'] ==
    1){ echo "<td>Read Uncommited</td>"; }
12.         else if ($block['transaction_isolation_leve
    l'] == 2){ echo "<td>Read Commited</td>"; }
13.         else if ($block['transaction_isolation_leve
    l'] == 3){ echo "<td>Repeatable Read</td>"; }
14.         else if ($block['transaction_isolation_leve
    l'] == 4){ echo "<td>Snapshot</td>"; }
```

```
15.          else if ($block['transaction_isolation_leve
     l'] == 5){ echo "<td>Serializable</td>"; }
16.      echo "<td>".$block['wait_type']."</td>";
17.      echo "<td>".$block['wait_time']."</td>";
18.      echo "<td>".$block['wait_resource']."</td>";
19.      echo "<td>".$block['query_text']."</td>";
20.    }
21. }
```

Gambar 5.33 Potongan fungsi untuk memanggil data historis *Blocking*

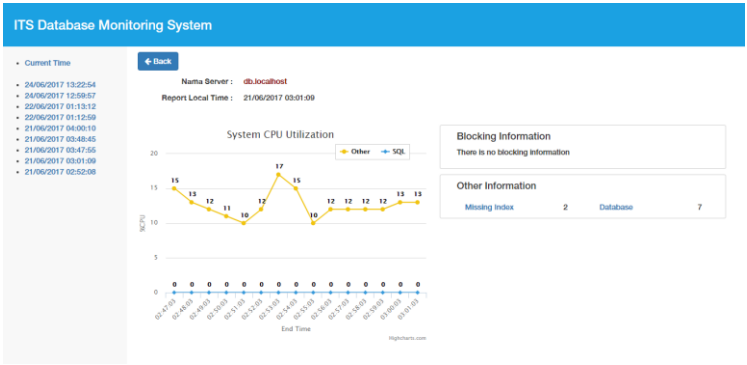
Setelah data historis *blocking* diambil dari tabel history_blocking, selanjutnya program akan menampilkan data tersebut ke dalam bentuk tabel. Potongan kode program untuk menampilkan data ke dalam bentuk tabel ditunjukkan pada Gambar 5.34 berikut ini.

```
1.   <div class="col-md-12 table-
     responsive" style="margin-top:20px;">
2.     <table class="table table-bordered">
3.       <thead style="font-size:10pt; background-
     color:#e8e8e8;">
4.         <tr>
5.           <th style="width: 100px;">Session ID : Requ
     est ID</th>
6.   <th style="width: 100px;">Blocking Session ID</th>
7.           <th>Status</th>
8.           <th>Transaction Name</th>
9.           <th>Isolation Level</th>
10.          <th>Wait Type</th>
11.          <th>Wait Time (ms)</th>
12.          <th>Wait Resource</th>
13.          <th>Query Text</th>
14.        </tr>
15.      </thead>
16.      <tbody style="font-size:9pt;">
17.  <?php get_history_blocking_info($db,$id_time,$id);
     ?>
18.      </tbody>
19.    </table> </div>
```

Gambar 5.34 Potongan his_blocking_info.php untuk menampilkan hasil fungsi dalam bentuk tabel

5.4.8. Fungsi menampilkan informasi historis *Missing Index*

Secara umum, tampilan antarmuka *dashboard* ketika admin ingin melihat data historis *missing index* juga tidak jauh berbeda dengan antarmuka *dashboard* ketika *missing* index terdeteksi pada server basis data secara *real-time* seperti pada Gambar 5.35 berikut ini.



Gambar 5.35 Tampilan antarmuka *dashboard* historis jika terdapat *Missing Index*

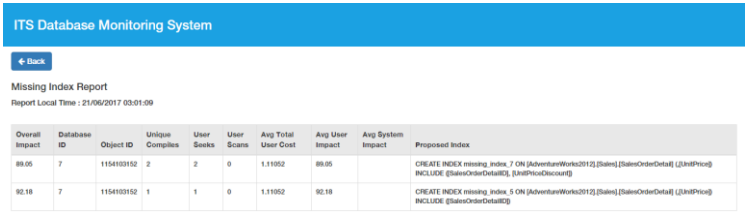
Pertama-tama, setelah program berhasil melakukan koneksi ke server DMBS, program akan mengambil data historis *missing index* dari tabel history_missing_index untuk ditampilkan pada antarmuka *dashboard* seperti yang ditunjukkan pada Gambar 5.36 berikut ini.

```
1.  <div class="col-md-6" style="padding-
    top: 10px; padding-bottom:10px;">
2.    <div class="col-md-10">
3.      <?php echo "<a href=\"history/his_missing_index
    .php?time={$id_time}&id={$id}\">Missing Index</a>"?
    >
4.    </div>
5.    <div class="col-md-2">
6.      <?php
```

```
7.          $sql4 = mysqli_query($db, "SELECT * FROM hist
    ory_missing_index WHERE cron_date = '$id_time' AND
    nama_sistem = '$id'");
8.          $row_count = mysqli_num_rows($sql4);
9.          echo $row_count;
10.       ?>
11.     </div>
12.  </div>
```

Gambar 5.36 Potongan history_server.php untuk menampilkan data *Missing Index*

Untuk melihat detail informasi historis ketika *missing index* terdeteksi pada server basis data, administrator dapat melakukan klik pada menu Missing Index. Kemudian sistem akan menampilkan halaman detail informasi *missing index* seperti pada Gambar 5.37 berikut ini.



Gambar 5.37 Tampilan antarmuka detail informasi historis untuk *Missing Index*

Untuk melakukan pengambilan data, program akan menjalankan fungsi get_history_missing_index_info() yang akan mengambil data dari tabel history_missing_index. Gambar 5.38 merupakan potongan kode program untuk memanggil fungsi tersebut.

```
1.  function get_history_missing_index_info($db,$id_tim
    e,$id){
2.    $sql = "SELECT * FROM history_missing_index WHERE
     cron_date = '$id_time' AND nama_sistem = '$id'";
3.    $res = mysqli_query($db, $sql) or die(mysqli_erro
    r($db));
4.    while ($miss_index = mysqli_fetch_array($res)){
```

```
5.      $pi01 = $miss_index['index_handle'];
6.      $pi02 = $miss_index['fully_qualified_object'];

7.      $pi03 = $miss_index['equality_columns'];
8.      $pi04 = $miss_index['inequality_columns'];
9.      $pi05 = str_replace(".",",",$miss_index['includ
    ed_columns']);
10.
11.     echo "<tr>";
12.     echo "<td>".$miss_index['database_id']."</td>";

13.     echo "<td>".$miss_index['object_id']."</td>";
14.     echo "<td>".$miss_index['unique_compiles']."</t
    d>";
15.     echo "<td>".$miss_index['user_seeks']."</td>";
16.     echo "<td>".$miss_index['user_scans']."</td>";
17.     echo "<td>".$miss_index['avg_total_user_cost'].
    "</td>";
18.     echo "<td>".$miss_index['avg_user_impact']."</t
    d>";
19.     echo "<td>".$miss_index['avg_system_impact']."<
    /td>";
20.       if(is_null($pi03) && is_null($pi04) && is_nul
    l($pi05)){
21.         echo "<td>CREATE INDEX missing_index_".$pi0
    1." ON ".$pi02."</td>";
22.       } else if (is_null($pi04) && is_null($pi05)){
23.         echo "<td>CREATE INDEX missing_index_".$pi0
    1." ON ".$pi02." (".$pi03.")</td>";
24.       } else if (is_null($pi03) && is_null($pi05)){
25.         echo "<td>CREATE INDEX missing_index_".$pi0
    1." ON ".$pi02." (".$pi04.")</td>";
26.       } else if (is_null($pi05)){
27.         echo "<td>CREATE INDEX missing_index_".$pi0
    1." ON ".$pi02." (".$pi03.",".$pi04.")</td>";
28.       } else if (is_null($pi04)){
29.         echo "<td>CREATE INDEX missing_index_".$pi0
    1." ON ".$pi02." (".$pi03.") INCLUDE (".$pi05.")</t
    d>";
30.       } else if (is_null($pi03)){
31.         echo "<td>CREATE INDEX missing_index_".$pi0
    1." ON ".$pi02." (".$pi04.") INCLUDE (".$pi05.")</t
    d>";
32.       } else {
```

```
33.          echo "<td>CREATE INDEX missing_index_".$pi0
    1." ON ".$pi02." (".$pi03.",".$pi04.") INCLUDE (".$
    pi05.")</td>";
34.        }
35.        echo "</tr>";
36.    }
37. }
```

Gambar 5.38 Potongan fungsi untuk memanggil data historis *Missing Index*

Setelah data historis *missing index* diambil dari tabel history_missing_index, selanjutnya program akan menampilkan data tersebut ke dalam bentuk tabel. Potongan kode program untuk menampilkan data ke dalam bentuk tabel ditunjukkan pada Gambar 5.39 berikut ini.

```
1.  <div class="col-md-12 table-
    responsive" style="margin-top:20px;">
2.    <table class="table table-bordered">
3.      <thead style="font-size:10pt; background-
    color:#e8e8e8;">
4.        <tr>
5.          <th>Database ID</th>
6.          <th>Object ID</th>
7.          <th>Unique Compiles</th>
8.          <th>User Seeks</th>
9.          <th>User Scans</th>
10.         <th>Avg Total User Cost</th>
11.         <th>Avg User Impact</th>
12.         <th>Avg System Impact</th>
13.         <th>Proposed Index</th>
14.       </tr>
15.     </thead>
16.     <tbody style="font-size:9pt;">
17.       <?php get_history_missing_index_info($db,$id_
    time,$id); ?>
18.     </tbody>
19.   </table>
20. </div>
```

Gambar 5.39 Potongan his_missing_index.php untuk menampilkan hasil fungsi dalam bentuk tabel

5.5. Pengujian Aplikasi

Pada pengerjaan tugas akhir ini, penulis melakukan dua macam pengujian aplikasi, yaitu *System Testing* dan *Version Testing.* Pada bagian ini akan dijelaskan masing-masing jenis pengujian yang dilakukan.

5.5.1. *System Testing*

System Testing merupakan pengujian aplikasi secara keseluruhan untuk menguji apakah aplikasi telah memenuhi spesifikasi sesuai kebutuhan yang dijelaskan pada bagian sebelumnya, memastikan implementasi desain dalam bentuk kode program berjalan dengan baik, dan memastikan tidak ada *bug* pada masing-masing fungsi

5.5.2. *Version Testing*

Version Testing merupakan pengujian kompabilitas aplikasi yang dibuat terhadap versi SQL Server yang lainnya. Pada pengerjaan tugas akhir ini, penulis menggunakan SQL Server 2012 sebagai server untuk basis data dan SQL Server Performance Dashboard 2012 untuk reporting. Berdasarkan petunjuk pada website Microsoft.com [18], terdapat beberapa versi SQL Server dan SQL Server Performance Dashboard sesuai kompabilitasnya untuk dilakukan pengujian versi pada sistem yang dibuat. Beberapa versi tersebut dijabarkan seperti pada Tabel 5.3 berikut ini.

Tabel 5.3 Tabel versi SQL Server dan Performance Dashboard yang akan dilakukan *Version Testing*

No.	DBMS	Performance Dashboard
1.	SQL Server 2005 SP2 – 9.00.3042.00 (32-bit)	Performance Dashboard 2005
2.	SQL Server 2008 R2 SP2 – 10.50.4000.0 (64-bit)	Performance Dashboard 2012
3.	SQL Server 2014 - 12.0.2000.8 (64-bit)	Performance Dashboard 2012

Halaman ini sengaja dikosongkan

BAB VI
HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan hasil dari pengujian aplikasi serta pembahasan terhadap hasil pengujian yang dilakukan. Berikut adalah penjelasan untuk hasil dan pembahasan pengujian yang dilakukan

6.1. Hasil

Pada subbab ini akan dijelaskan mengenai hasil pengujian dari implementasi kode program pada sistem yang telah dijelaskan pada bab sebelumnya. Terdapat dua pengujian yaitu *system testing* dan *version testing* yang akan dibahas pada bagian ini.

6.1.1. *System Testing*

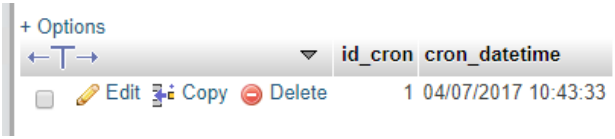
System Testing akan melakukan pengujian terhadap fungsi-fungsi sistem apakah sudah berjalan dengan baik atau masih terdapat *bug* pada sistem. Tabel 6.1 menunjukan fungsi atau fitur yang akan dilakukan pengujian.

Tabel 6.1 Fitur pada sistem yang akan diuji

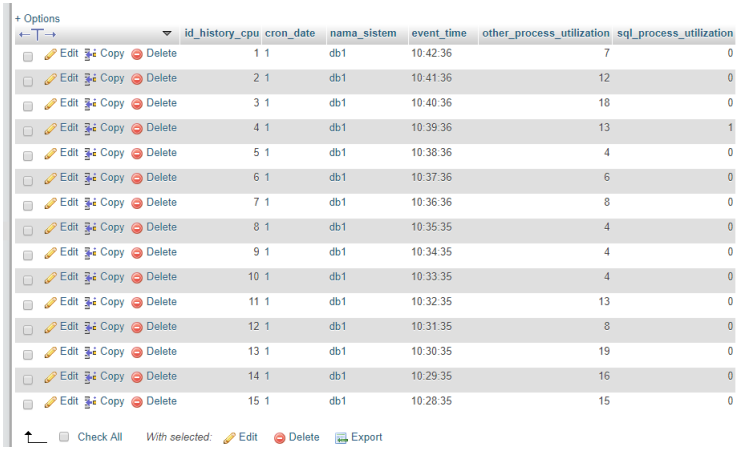
No.	Fitur yang diuji
1.	*Crawler*
2.	Informasi kondisi server
3.	Informasi metrik *CPU Utilization*
4.	Informasi metrik *Blocking*
5.	Informasi metrik *Missing Index*
6.	Informasi historis metrik *CPU Utilization*
7.	Informasi historis metrik *Blocking*
8.	Informasi historis metrik *Missing Index*

6.1.1.1. Crawler

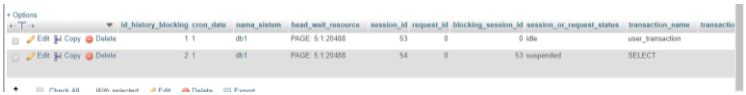
Pada fungsi ini, pertama-tama dilakukan pengujian untuk koneksi ke server basis data. Apabila koneksi tidak berhasil maka tidak ada data historis yang diambil dan program akan menyimpan data waktu *crawling* pada basis data. Apabila koneksi berhasil dilakukan ke semua server basis data, maka program akan menjalankan *stored procedure* dan menyimpan data kondisi server basis data sesuai keadaan waktu proses *crawling* dijalankan.



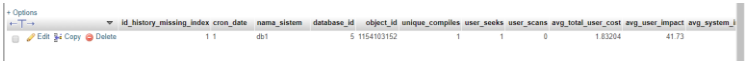
Gambar 6.1 Data waktu *crawling* berhasil diinputkan



Gambar 6.2 Data historis metrik *CPU Utilization* berhasil diinputkan



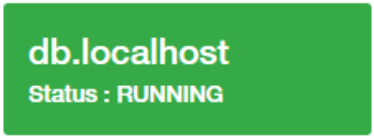
Gambar 6.3 Data historis metrik *Blocking* berhasil diinputkan



Gambar 6.4 Data historis metrik *Missing Index* berhasil diinputkan

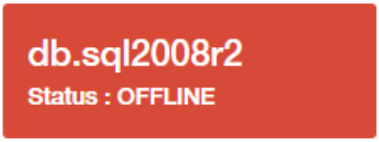
6.1.1.2. Informasi kondisi server

Pada saat sistem pertama kali dijalankan. Tampilan antarmuka yang terlihat adalah kondisi server basis data saat ini. Terdapat 3(tiga) kategori yang diimplementasikan pada sistem ini. Jika kondisi server berjalan dengan baik, maka sistem akan menampilkan keterangan server dengan warna hijau seperti pada Gambar 6.5 berikut



Gambar 6.5 Kondisi server berjalan dengan baik

Namun, jika kondisi server sedang dalam keadaan *offline* atau sistem tidak dapat melakukan koneksi ke server, maka sistem akan menampilkan keterangan server dengan warna merah seperti pada Gambar 6.6 berikut



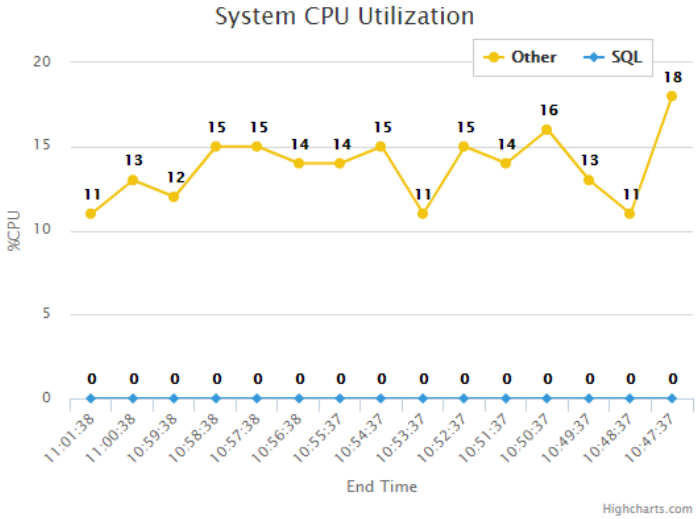
Gambar 6.6 Kondisi server sedang *offline*

Dan jika kondisi server sedang terjadi *blocking* maka sistem akan menampilkan keterangan server dengan warna merah seperti pada Gambar 6.7 berikut

Gambar 6.7 Kondisi server saat terjadi *blocking*

6.1.1.3. Informasi metrik *CPU Utilization*

Informasi metrik *CPU Utilization* pada sistem ditampilkan dalam chart pada *dashboard*. Tampilan antarmuka *dashboard* saat admin melakukan klik pada salah satu server ditunjukkan pada Gambar 6.8 berikut ini.



Gambar 6.8 Tampilan informasi metrik *CPU Utilization*

Namun, pada gambar terlihat kesalahan pembacaan data. Pada Gambar 6.8 terlihat informasi waktu pada chart ditampilkan

dari kanan ke kiri, yang seharusnya ditampilkan dari kiri ke kanan. Hal ini disebabkan karena *query* yang digunakan pada *stored procedure* bawaan dari SQL Server Performance Dashboard seperti pada Gambar 6.9 berikut ini.

```
1.  ALTER procedure [MS_PerfDashboard].[usp_Main_GetCPU
    History]
2.  AS
3.  BEGIN
4.    DECLARE @ms_now bigint
5.    SELECT @ms_now = ms_ticks FROM sys.dm_os_sys_info
    ;
6.    SELECT TOP 15 record_id,
7.      dateadd(ms, -
    1 * (@ms_now - [timestamp]), GetDate()) AS EventTim
    e,
8.      SQLProcessUtilization,SystemIdle,
9.      100 - SystemIdle - SQLProcessUtilization AS Oth
    erProcessUtilization
10.   FROM (
11.     SELECT
12.       record.value('(./Record/@id)[1]', 'int') AS r
    ecord_id,
13.       record.value('(./Record/SchedulerMonitorEvent
    /SystemHealth/SystemIdle)[1]', 'int') AS SystemIdle
    ,
14.       record.value('(./Record/SchedulerMonitorEvent
    /SystemHealth/ProcessUtilization)[1]', 'int') AS SQ
    LProcessUtilization,
15.       timestamp
16.     FROM (
17.       SELECT timestamp, convert(xml, record) AS rec
    ord
18.       FROM sys.dm_os_ring_buffers
19.       WHERE ring_buffer_type = N'RING_BUFFER_SCHEDU
    LER_MONITOR'
20.       AND record LIKE '%<SystemHealth>%') AS x
21.     ) AS y
22.     ORDER BY record_id DESC
23. END
```

Gambar 6.9 *Query* awal pada *stored procedure*

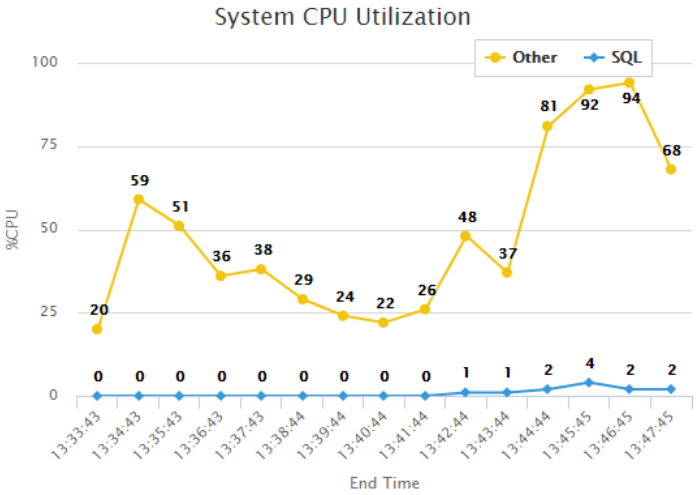
Oleh karena itu, dilakukan perubahan *query* pada *stored procedure* tersebut seperti pada gambar 6.10 agar informasi pada chart ditampilkan dengan benar.

```
1.  ALTER procedure [MS_PerfDashboard].[usp_Main_GetCPU
    History]
2.  AS
3.  BEGIN
4.  DECLARE @ms_now bigint
5.  SELECT @ms_now = ms_ticks FROM sys.dm_os_sys_info;

6.  SELECT * FROM (
7.    SELECT TOP 15 record_id,
8.      dateadd(ms, -
    1*(@ms_now - [timestamp]), GetDate()) as EventTime,
9.      SQLProcessUtilization,SystemIdle,
10.     100 - SystemIdle - SQLProcessUtilization as Oth
    erProcessUtilization
11.   FROM (
12.     SELECT
13.       record.value('(./Record/@id)[1]', 'int') AS r
    ecord_id,
14.      record.value('(./Record/SchedulerMonitorEvent/
    SystemHealth/SystemIdle)[1]', 'int') AS SystemIdle,
15.       record.value('(./Record/SchedulerMonitorEvent
    /SystemHealth/ProcessUtilization)[1]', 'int') AS SQ
    LProcessUtilization, timestamp
16.     FROM (
17.   SELECT timestamp, convert(xml, record) AS record
18.       FROM sys.dm_os_ring_buffers
19.       WHERE ring_buffer_type = N'RING_BUFFER_SCHEDU
    LER_MONITOR'
20.       AND record LIKE '%<SystemHealth>%'
21.       ) AS x
22.     ) as y
23.     ORDER BY record_id DESC
24.     ) AS z
25.   ORDER BY record_id ASC
26. END
```

Gambar 6.10 *Query* pada *stored procedure* setelah dilakukan perubahan

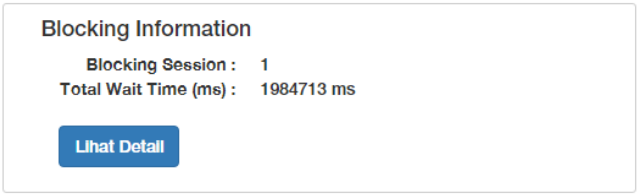
Setelah dilakukan perubahan pada *query stored procedure* tersebut, maka informasi yang ditampilkan pada chart menjadi seperti pada Gambar 6.11 berikut.



Gambar 6.11 Tampilan metrik *CPU Utilization* setelah dilakukan perubahan *query*

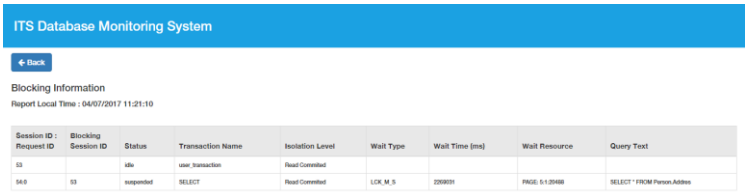
6.1.1.4. Informasi metrik *Blocking*

Jika terdapat *blocking* pada server basis data, informasi tersebut dapat ditampilkan pada *dashboard* seperti pada Gambar 6.12.



Gambar 6.12 Tampilan antarmuka informasi *blocking* pada *dashboard*

Jika admin, ingin melihat detail informasi mengenai *blocking* pada server basis data, admin dapat melakukan klik pada tombol Lihat Detail. Kemudian sistem menampilkan halaman antarmuka informasi detail *Blocking* seperti pada Gambar 6.13 berikut ini.



Gambar 6.13 Tampilan antarmuka detail informasi *blocking*

Apabila tidak terjadi *blocking* pada server basis data, sistem akan menampilkan keterangan pada *dashboard* seperti pada Gambar 6.14 berikut ini.

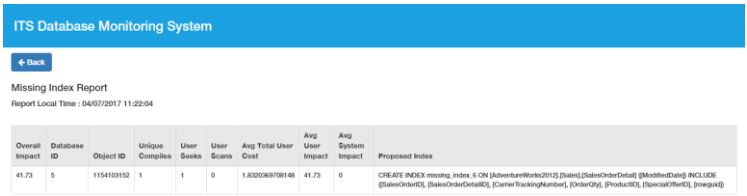
Gambar 6.14 Tampilan antarmuka jika tidak terjadi *blocking*

6.1.1.5. Informasi metrik *Missing Index*

Informasi *missing index* yang terdeteksi oleh sistem pada server basis data ditampilkan pada *dashboard* seperti pada Gambar 6.15 berikut ini.

Gambar 6.15 Tampilan antarmuka informasi *missing index* pada *dashboard*

Jika admin, ingin melihat detail informasi mengenai *missing index* yang terdeteksi pada server basis data, admin dapat melakukan klik pada link dengan tulisan Missing Index. Kemudian sistem menampilkan halaman antarmuka informasi detail *Missing Index* seperti pada Gambar 6.16 berikut ini.

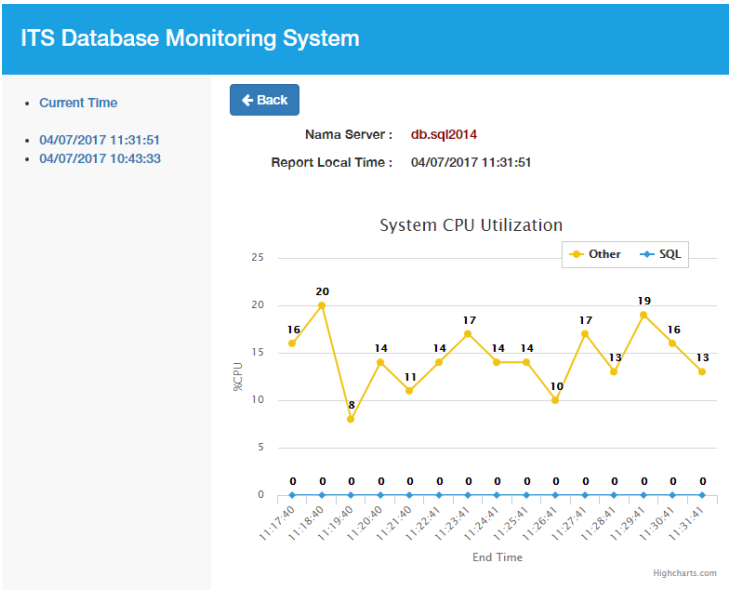


Gambar 6.16 Tampilan antarmuka detail informasi *missing index*

6.1.1.6. Informasi historis metrik *CPU Utilization*

Informasi historis *CPU Utilization* yang ditampilkan pada *dashboard* secara keseluruhan menyerupai antarmuka *dashboard* informasi *CPU* secara *real-time* seperti pada Gambar 6.17.

Untuk menampilkan halaman historis *CPU Utilization*, admin melakukan klik pada salah satu waktu yang ingin dilihat pada menu sebelah kiri. Kemudian sistem akan menampilkan informasi historis berdasarkan waktu yang dipilih oleh admin. Informasi historis didapatkan melalui proses *crawling* yang disimpan pada basis data situs pantau.its.ac.id



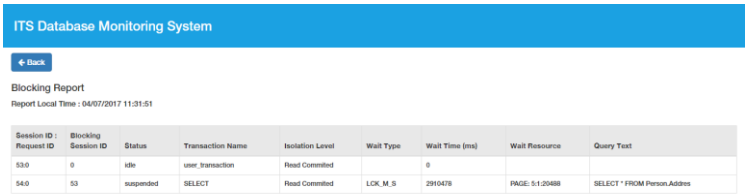
Gambar 6.17 Tampilan antarmuka informasi historis *CPU Utilization* pada *dashboard*

6.1.1.7. Informasi historis metrik *Blocking*

Informasi historis *blocking* yang terjadi pada server basis data yang ditampilkan pada *dashboard* secara keseluruhan menyerupai antarmuka *dashboard* informasi *Blocking* yang terjadi secara *real-time* seperti pada Gambar 6.18 berikut ini.

Gambar 6.18 Tampilan antarmuka informasi historis *Blocking* pada *dashboard*

Jika admin, ingin melihat detail informasi historis mengenai *blocking*, admin dapat melakukan klik pada link dengan tulisan Missing Index. Kemudian sistem menampilkan halaman antarmuka informasi detail historis *Blocking* seperti pada Gambar 6.19 berikut ini.



Gambar 6.19 Tampilan antarmuka detail informasi historis *Blocking*

6.1.1.8. Informasi historis metrik *Missing Index*

Informasi historis *missing index* yang terdeteksi oleh sistem pada server basis data yang ditampilkan pada *dashboard* secara keseluruhan menyerupai antarmuka *dashboard* informasi *Missing Index* yang terdeteksi secara *real-time* seperti pada Gambar 6.20 berikut ini.

Gambar 6.20 Tampilan antarmuka informasi historis *Missing Index* pada *dashboard*

Jika admin, ingin melihat detail informasi historis mengenai *missing index*, admin dapat melakukan klik pada link dengan tulisan Missing Index. Kemudian sistem menampilkan halaman antarmuka informasi detail historis *Missing Index* seperti pada Gambar 6.21 berikut ini.

Gambar 6.21 Tampilan antarmuka detail informasi historis *Missing Index*

6.1.2. *Version Testing*

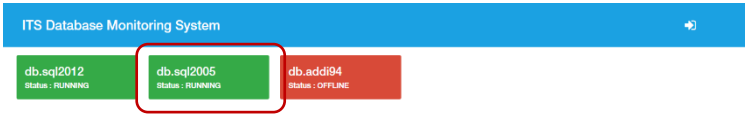
Version Testing merupakan pengujian kompabilitas aplikasi yang dibuat terhadap versi SQL Server yang lainnya. Pada pengerjaan tugas akhir ini, penulis menggunakan SQL Server 2012 sebagai server untuk basis data dan SQL Server Performance Dashboard 2012 untuk reporting. Berdasarkan petunjuk pada website Microsoft.com [18], terdapat beberapa versi SQL Server dan SQL Server Performance Dashboard sesuai kompabilitasnya untuk dilakukan pengujian versi pada sistem yang dibuat. Beberapa versi yang akan dilakukan *version testing* dijabarkan seperti pada Tabel 6.2 berikut ini.

Tabel 6.2 Daftar versi yang akan dilakukan *Version Testing*

No.	SQL Server	Performance Dashboard
1.	Microsoft SQL Server 2005 SP2 – 9.00.3042.00 (32-bit)	SQL Server Performance Dashboard 2005
2.	Microsoft SQL Server 2008 R2 SP2 – 10.50.4000.0 (64-bit)	SQL Server Performance Dashboard 2012
3.	Microsoft SQL Server 2014 - 12.0.2000.8 (64-bit)	SQL Server Performance Dashboard 2012

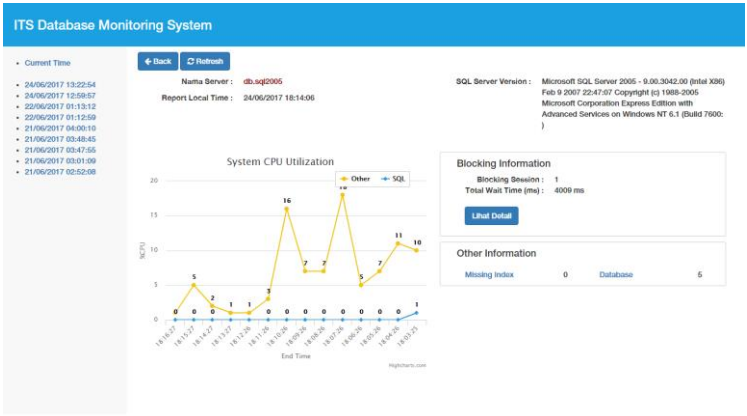
6.1.2.1. SQL Server 2005 dan SQL Server Performance Dashboard 2005

Pertama-tama dilakukan pengujian terhadap koneksi ke server basis data yang menggunakan SQL Server 2005. Pada Gambar 6.22 terlihat bahwa koneksi berhasil dilakukan dan menampilkan informasi bahwa server berjalan dengan baik.



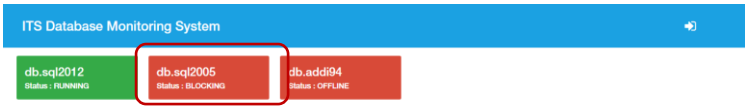
Gambar 6.22 Kondisi server dengan SQL Server 2005 berjalan dengan baik

Kemudian dilakukan pengujian untuk halaman antarmuka *dashboard*. Pada Gambar 6.23 menunjukkan bahwa halaman *dashboard* berhasil ditampilkan dengan baik.



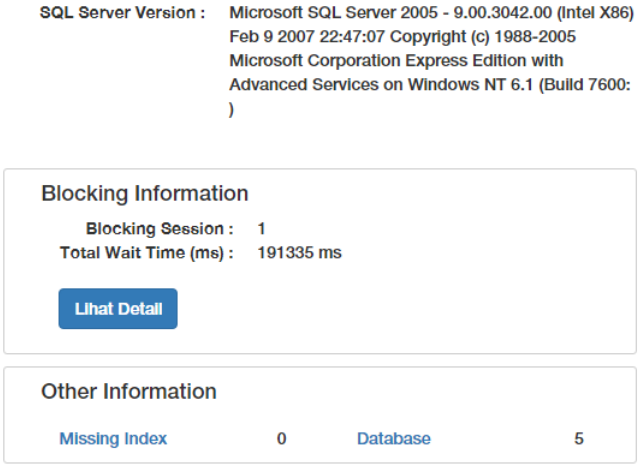
Gambar 6.23 Tampilan antarmuka *dashboard* pada server SQL Server 2005

Selanjutnya pengujian untuk metrik *Blocking*. Pada Gambar 6.24 terlihat sistem dapat menampilkan keterangan apabila sedang terjadi *blocking* pada server.



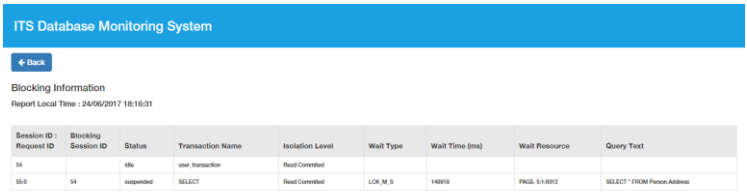
Gambar 6.24 Kondisi server SQL Server 2005 saat terjadi *blocking*

Pengujian selanjutnya yaitu sistem menampilkan antarmuka *dashboard* ketika server sedang terjadi *blocking.* Pada Gambar 6.25 terlihat bahwa halaman antarmuka *dashboard* dapat ditampilkan dengan baik.



Gambar 6.25 Tampilan antarmuka *dashboard Blocking* pada SQL Server 2005

Kemudian dilakukan pengujian apakah sistem mampu untuk menampilkan halaman detail informasi *Blocking* pada SQL Server 2005. Pada Gambar 6.26 terlihat bahwa sistem dapat menampilkan halaman informasi detail *Blocking* dengan baik.

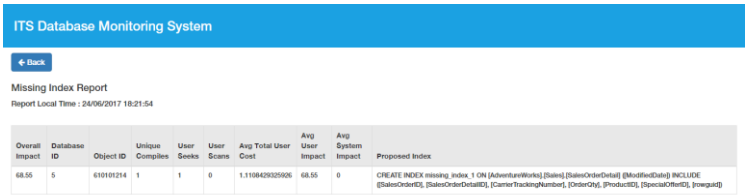


Gambar 6.26 Tampilan antarmuka informasi detail *Blocking* pada SQL Server 2005

Proses terakhir adalah pengujian untuk metrik *Missing Index*. Pada Gambar 6.27 jika *missing index* terdeteksi pada server SQL Server 2005, maka sistem dapat menampilkan informasi *missing index* pada *dashboard.*

Gambar 6.27 Tampilan antarmuka *dashboard Missing Index* pada SQL Server 2005

Kemudian dilakukan pengujian apakah sistem dapat menampilkan halaman informasi detail *missing index* yang terdeteksi pada server SQL Server 2005. Gambar 6.28 menunjukkan sistem dapat menampilkan halaman informasi detail *missing index* dengan baik.

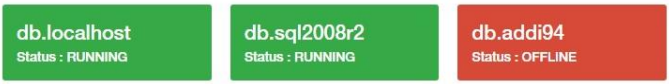


Gambar 6.28 Tampilan antarmuka informasi detail *Missing Index* pada SQL Server 2005

Dengan demikian maka sistem dapat berfungsi dengan baik pada server yang menggunakan versi SQL Server 2005 dan terinstal SQL Server Performance Dashboard 2005.

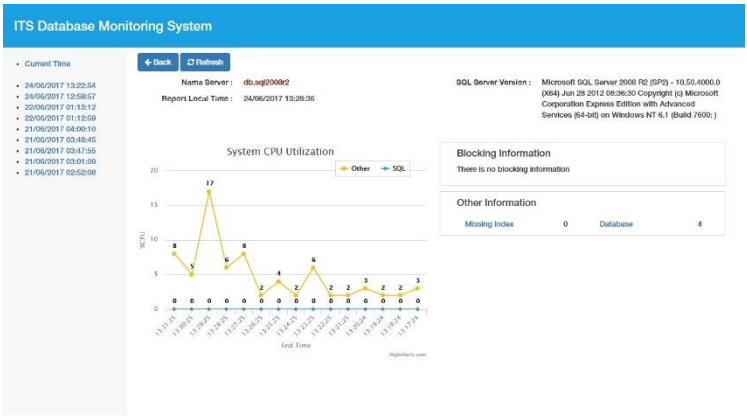
6.1.2.2. SQL Server 2008 R2 dan SQL Server Performance Dashboard 2012

Pertama-tama dilakukan pengujian terhadap koneksi ke server basis data yang menggunakan SQL Server 2008 R2. Pada Gambar 6.29 terlihat bahwa koneksi berhasil dilakukan dan menampilkan informasi bahwa server berjalan dengan baik.



Gambar 6.29 Kondisi server dengan SQL Server 2008 R2 berjalan dengan baik

Kemudian dilakukan pengujian untuk halaman antarmuka *dashboard*. Pada Gambar 6.30 menunjukkan bahwa halaman *dashboard* berhasil ditampilkan dengan baik.

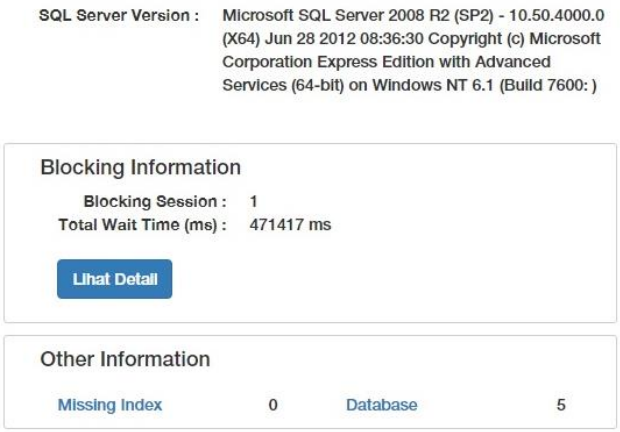


Gambar 6.30 Tampilan antarmuka *dashboard* pada server SQL Server 2008 R2

Selanjutnya pengujian untuk metrik *Blocking.* Pada Gambar 6.31 terlihat sistem dapat menampilkan keterangan apabila sedang terjadi *blocking* pada server.

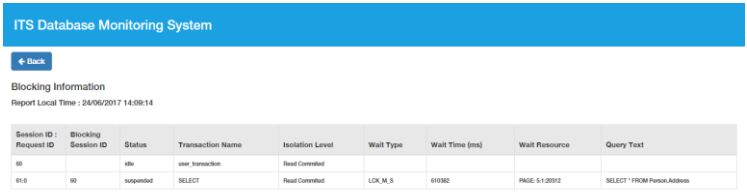
Gambar 6.31 Kondisi server SQL Server 2008 R2 saat terjadi *blocking*

Pengujian selanjutnya yaitu sistem menampilkan antarmuka *dashboard* ketika server sedang terjadi *blocking.* Pada Gambar 6.32 terlihat bahwa halaman antarmuka *dashboard* dapat ditampilkan dengan baik.



Gambar 6.32 Tampilan antarmuka *dashboard Blocking* pada SQL Server 2008 R2

Kemudian dilakukan pengujian apakah sistem mampu untuk menampilkan halaman detail informasi *Blocking* pada SQL Server 2008 R2. Pada Gambar 6.33 terlihat bahwa sistem dapat menampilkan halaman informasi detail *Blocking* dengan baik.

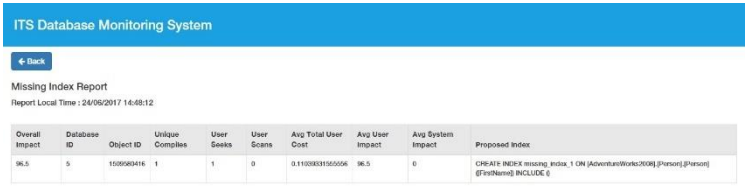


Gambar 6.33 Tampilan antarmuka informasi detail *Blocking* pada SQL Server 2008 R2

Proses terakhir adalah pengujian untuk metrik *Missing Index*. Pada Gambar 6.34 jika *missing index* terdeteksi pada server SQL Server 2008 R2, maka sistem dapat menampilkan informasi *missing index* pada *dashboard.*

Gambar 6.34 Tampilan antarmuka *dashboard Missing Index* pada SQL Server 2008 R2

Kemudian dilakukan pengujian apakah sistem dapat menampilkan halaman informasi detail *missing index* yang terdeteksi pada server SQL Server 2008 R2. Gambar 6.35 menunjukkan sistem dapat menampilkan halaman informasi detail *missing index* dengan baik.

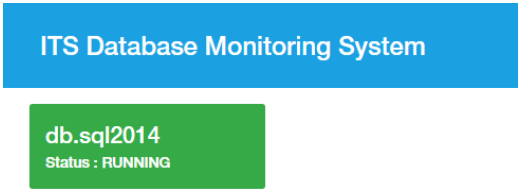


Gambar 6.35 Tampilan antarmuka informasi detail *Missing Index* pada SQL Server 2008 R2

Dengan demikian maka sistem dapat berfungsi dengan baik pada server yang menggunakan versi SQL Server 2008 R2 dan terinstal SQL Server Performance Dashboard 2012.

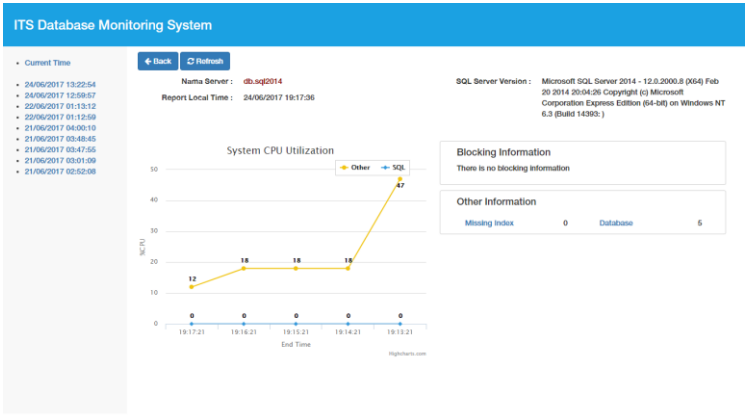
6.1.2.3. SQL Server 2014 dan SQL Server Performance Dashboard 2012

Pertama-tama dilakukan pengujian terhadap koneksi ke server basis data yang menggunakan SQL Server 2014. Pada Gambar 6.36 terlihat bahwa koneksi berhasil dilakukan dan menampilkan informasi bahwa server berjalan dengan baik.



Gambar 6.36 Kondisi server dengan SQL Server 2014 berjalan dengan baik

Kemudian dilakukan pengujian untuk halaman antarmuka *dashboard*. Pada Gambar 6.37 menunjukkan bahwa halaman *dashboard* berhasil ditampilkan dengan baik.

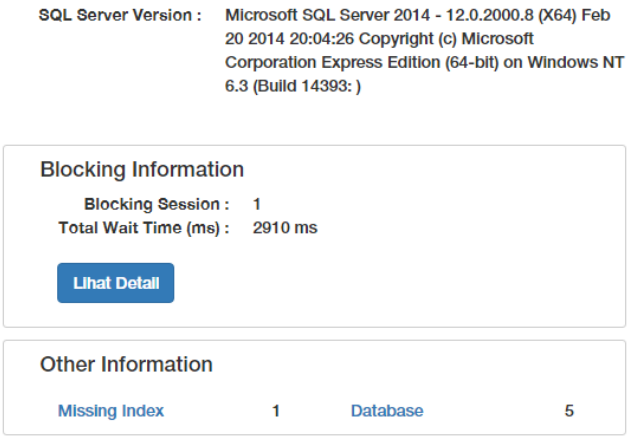


Gambar 6.37 Tampilan antarmuka *dashboard* pada server SQL Server 2014

Selanjutnya pengujian untuk metrik *Blocking*. Pada Gambar 6.38 terlihat sistem dapat menampilkan keterangan apabila sedang terjadi *blocking* pada server.

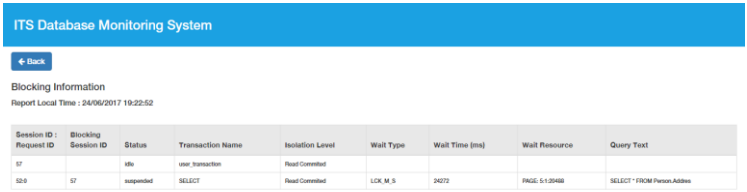
Gambar 6.38 Kondisi server SQL Server 2014 saat terjadi *blocking*

Pengujian selanjutnya yaitu sistem menampilkan antarmuka *dashboard* ketika server sedang terjadi *blocking*. Pada Gambar 6.39 terlihat bahwa halaman antarmuka *dashboard* dapat ditampilkan dengan baik.



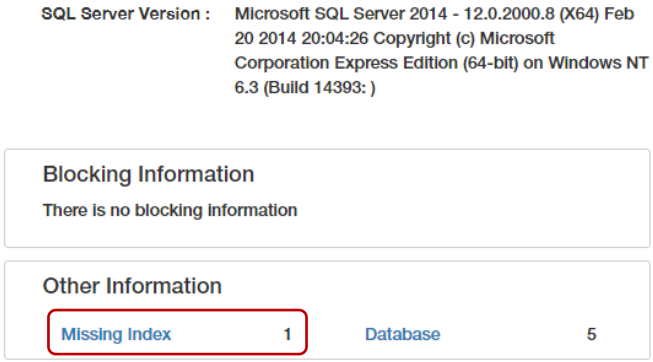
Gambar 6.39 Tampilan antarmuka *dashboard Blocking* pada SQL Server 2014

Kemudian dilakukan pengujian apakah sistem mampu untuk menampilkan halaman detail informasi *Blocking* pada SQL Server 2014. Pada Gambar 6.40 terlihat bahwa sistem dapat menampilkan halaman informasi detail *Blocking* dengan baik.



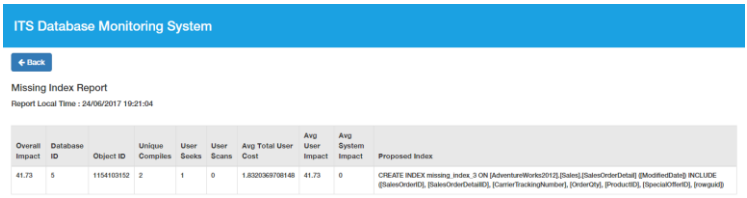
Gambar 6.40 Tampilan antarmuka informasi detail *Blocking* pada SQL Server 2014

Proses terakhir adalah pengujian untuk metrik *Missing Index*. Pada Gambar 6.41 jika *missing index* terdeteksi pada server SQL Server 2014, maka sistem dapat menampilkan informasi *missing index* pada *dashboard*.



Gambar 6.41 Tampilan antarmuka *dashboard Missing Index* pada SQL Server 2014

Kemudian dilakukan pengujian apakah sistem dapat menampilkan halaman informasi detail *missing index* yang terdeteksi pada server SQL Server 2014. Gambar 6.42 menunjukkan sistem dapat menampilkan halaman informasi detail *missing index* dengan baik.



Gambar 6.42 Tampilan antarmuka informasi detail *Missing Index* pada SQL Server 2014

Dengan demikian maka sistem dapat berfungsi dengan baik pada server yang menggunakan versi SQL Server 2014 dan terinstal SQL Server Performance Dashboard 2012.

6.2. Pembahasan

Pada subbab ini akan dijelaskan mengenai pembahasan terkait hasil pengujian yang telah dilakukan pada bagian sebelumnya.

6.2.1. Pembahasan *System Testing*

Pada saat dilakukan *System Testing*, setiap fungsi dari sistem telah berjalan dengan baik dan mampu dioperasikan sesuai dengan *output* yang diharapkan. Dari hasil *System Testing* yang telah dilakukan, ditemukan adanya kesalahan penyampaian informasi pada chart akibat *query* pada *stored procedure*. Kemudian dilakukan perubahan terhadap *query* agar menghasilkan *output* sesuai yang diharapkan. Dengan telah dilakukannya *system testing* secara keseluruhan pada sistem, maka sistem dapat disimpulkan telah berjalan dengan baik dan benar.

6.2.2. Pembahasan *Version Testing*

Pada saat dilakukan *Version Testing* terhadap berbagai macam versi SQL Server dan SQL Server Performance Dashboard, didapatkan hasil bahwa sistem mampu menyesuaikan terhadap

semua versi yang diujikan. Tidak ada pengaruh terhadap perbedaan versi yang digunakan oleh server basis data selama *stored procedure* yang dibutuhkan tersedia pada server basis data. *Stored procedure* tersebut diperlukan untuk menerima data pada setiap metrik SQL Server Performance Dashboard.

BAB VII
KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran dari pengerjaan tugas akhir ini. Kesimpulan dan saran yang ada diharapkan dapat berguna untuk proses pengembangan selanjutnya.

7.1. Kesimpulan

Berdasarkan dari hasil yang didapatkan pada pengerjaan tugas akhir, penulis menyimpulkan beberapa hal sebagai berikut:

1. Pembuatan tools pemantauan kinerja basis data berbasis web dengan memanfaatkan metrik-metrik pada SQL Server Performance Dashboard, yaitu *Blocking*, *Missing Index*, dan *CPU Utilization* dapat dilakukan dan diimplementasikan dengan baik pada sistem.
2. Metrik-metrik pada SQL Server Performance Dashboard diimplementasikan pada sistem berbasis web dengan cara memanggil *stored procedure* dari setiap metrik yang sudah tersedia saat menginstal SQL Server Performance Dashboard.
3. Berdasarkan hasil pengujian *System Testing*, semua fungsi pada sistem dapat berjalan dengan baik dan sesuai dengan *output* yang diharapkan.
4. Berdasarkan hasil pengujian *Version Testing*, tidak ada pengaruh terhadap perbedaan versi SQL Server dan SQL Server Performance Dashboard yang digunakan oleh server basis data, karena sistem hanya memanfaatkan *stored procedure* yang terdapat pada server basis data saat sudah terinstal SQL Server Performance Dashboard.

7.2. Saran

Untuk pengembangan sistem agar kedepannya lebih baik dan lebih bermanfaat, maka terdapat beberapa saran yang dapat dipertimbangkan yaitu:

1. Menambahkan metrik-metrik SQL Server Performance Dashboard yang lainnya agar fitur yang dimiliki sistem lebih lengkap dan lebih memudahkan admin dalam memantau kinerja basis data.
2. Menambahkan dan memperlengkapi sistem dengan halaman *back-end* untuk memudahkan admin dalam pengelolaan data server.

DAFTAR PUSTAKA

[1] "Database", en.wikipedia.org, 2017. [Online]. Available:https://en.wikipedia.org/wiki/Database. [Diakses: 01 Maret 2017].

[2] "Relational database", en.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Relational_database. [Diakses: 01 Maret 2017].

[3] Shee, Dek J. "*Database performance monitoring method and tool*." U.S. Patent No. 6,910,036. 21 Jun. 2005.

[4] S. Paul, "*Database Systems Performance Evaluation Technique*," 2008.

[5] "DB-Engines Ranking," [Online]. Available: http://db-engines.com/en/ranking. [Diakses: 20 Februari 2017].

[6] "Database Performance Monitoring," Technopedia, [Online]. Available:https://www.techopedia.com/definition/30560/database-performance-monitoring. [Diakses: 20 Februari 2017].

[7] "Enterprise Monitoring," Oracle, [Online]. Available: http://www.oracle.com/technetwork/oem/sys-mgmt/index.html.

[8] "SQL Server Performance Dashboard," Microsoft, [Online]. Available:https://sqldashboards.codeplex.com/.

[9] "Database Health," [Online]. Available: http://databasehealth.com/.

[10] "SQL Server Tools for DBA and Developers," ApexSQL, [Online]. Available: http://apexsql.com/.

[11] "ITS Web Monitoring System," [Online]. Available: http://pantau.its.ac.id.

[12] "ApexSQL Monitor - SQL Server monitoring tool | ApexSQL", Apexsql.com, 2017. [Online]. Available: http://www.apexsql.com/sql_tools_monitor.aspx. [Diakses: 02 Maret 2017].

[13] "What's next in ApexSQL Monitor 2017 - Web interface - Blog", Blog, 2017. [Online]. Available: http://blog.apexsql.com/whats-next-in-apexsql-monitor-2017-web-interface/. [Diakses: 02 Maret 2017].

[14] "Database Performance Analyzer SolarWinds", Solarwinds.com, 2017. [Online]. Available: http://www.solarwinds.com/database-performance-analyzer. [Diakses: 02 Maret 2017].

[15] T. Connolly and C. Begg, *Database Systems, A Practical Approach to Design, Implementation, and Management, 6th ed.* page 598. Pearson, 2015.

[16] *Microsoft SQL Server 2012 Performance Dashboard Reports Manual.* Microsoft

[17] S.Few, *Information Dashboard Design.* O'Reilly, 2006.

[18] "Download Microsoft SQL Server Performance Dashboard 2012", Microsoft, 2017. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=29063 [Diakses: 12 Juni 2017].

[19] K. Delaney, *Microsoft SQL Server 2012 Internals*. O'Reilly Media, Inc., 2013.

[20] "sys.dm_exec_sessions (Transact-SQL)", *Docs.microsoft.com*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-exec-sessions-transact-sql. [Diakses: 12 Juni 2017].

[21] "sys.dm_exec_requests (Transact-SQL)", *Docs.microsoft.com*, 2017. [Online]. Available:

https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-exec-requests-transact-sql. [Diakses: 12 Juni 2017].

[22] "sys.dm_tran_active_transactions (Transact-SQL)", *Docs.microsoft.com*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-tran-active-transactions-transact-sql. [Diakses: 12 Juni 2017].

[23] "sys.dm_tran_session_transactions (Transact-SQL)", *Docs.microsoft.com*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-tran-session-transactions-transact-sql. [Diakses: 12 Juni 2017].

[24] "sys.dm_db_missing_index_group_stats (Transact-SQL)", *Docs.microsoft.com*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-db-missing-index-group-stats-transact-sql. [Diakses: 12 Juni 2017].

[25] "sys.dm_db_missing_index_details (Transact-SQL)", *Docs.microsoft.com*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-db-missing-index-details-transact-sql. [Diakses 12 Juni 2017].

[26] "sys.dm_os_sys_info (Transact-SQL)", *Docs.microsoft.com*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-os-sys-info-transact-sql. [Diakses: 12 Juni 2017].

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Penulis lahir di Kota Banjarmasin pada tanggal 20 Desember 1994 ini merupakan anak pertama dari dua bersaudara. Penulis telah menyelesaikan pendidikan formal di SD Yayasan Hippindo Banjarmasin, SMP Negeri 6 Banjarmasin, SMA Negeri 7 Banjarmasin, dan akhirnya penulis masuk menjadi mahasiswa Jurusan Sistem Informasi ITS pada tahun 2012 melalui jalur SNMPTN Ujian Tertulis dan terdaftar dengan NRP 5212100094. Selama menempuh dunia perkuliahan, penulis aktif dalam organisasi dan kepanitian kegiatan kemahasiswaan, yaitu menjadi Staff Departemen Riset dan Pengembangan Teknologi (RTD) BEM FTIf ITS 2013/2014, Kepala Divisi Aplikasi Teknologi Departemen Riset dan Pengembangan Teknologi (RTD) BEM FTIf ITS 2014/2015, Staff Divisi Humas PMK ITS 2014/2015, dan Staff Divisi Persekutuan PMK ITS 2015/2016.

Pada tahun terakhir ini, penulis mengambil bidang minat penelitian Akusisi Data dan Diseminasi Informasi dengan konsentrasi Teknologi Basis Data. Penulis dapat dihubungi melalui email erwinmapaliey@gmail.com